S3F94C8/F94C4

8-Bit CMOS MICROCONTROLLERS

Revision 1.00

July 2010

USER'S MANUAL

SAMSUNG ELECTRONICS RESERVES THE RIGHT TO CHANGE PRODUCTS, INFORMATION AND SPECIFICATIONS WITHOUT NOTICE.

Products and specifications discussed herein are for reference purposes only. All information discussed herein is provided on an "AS IS" basis, without warranties of any kind.

This document and all information discussed herein remain the sole and exclusive property of Samsung Electronics. No license of any patent, copyright, mask work, trademark or any other intellectual property right is granted by one party to the other party under this document, by implication, estoppel or otherwise.

Samsung products are not intended for use in life support, critical care, medical, safety equipment, or similar applications where product failure could result in loss of life or personal or physical harm, or any military or defense application, or any governmental procurement to which special terms or provisions may apply.

For updates or additional information about Samsung products, contact your nearest Samsung office.

All brand names, trademarks and registered trademarks belong to their respective owners.

© 2010 Samsung Electronics Co., Ltd. All rights reserved.

SAMSUNG ELECTRONICS

SAMSUNG

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product.

Copyright © 2010 Samsung Electronics Co., Ltd.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electric or mechanical, by photocopying, recording, or otherwise, without the prior written consent of Samsung Electronics.

Samsung Electronics Co., Ltd.
San #24 Nongseo-Dong, Giheung-Gu
Yongin-City, Gyeonggi-Do, Korea 446-711

Contact Us: *younghee46.won@samsung.com*
TEL: (82)-(31)-209-3865
FAX: (82)-(31)-209-6494

Home Page: *http://www.samsungsemi.com*

Revision History

Revision No.	Date	Description	Remark
V0.0	Nov. 2008	• SAM8RC(modified) • 8K Full flash ROM • 208 Bytes RAM • Interrupt : 1 vector / 4 sources • I/O : 18(max) • PWM : 1 channel / 3 resolution • Timer : Basic timer and 8-bit timer 0 • ADC : 10-bit x 9 channel • Operating voltage: – 1.8-2.7V @ 1-4MHz – 2.7-5.5V @ 1-10MHz • Operating Temp : – 40°C to + 85°C • Internal RC tolerance : 1% (typ.) 3% (max) @– 40°C to + 85°C, 1.8-5.5V. • Stop current : 1uA(typ.) 2uA (max) @ 25°C, 4.5-5.5V	
V0.1	Dec.16, 2008	• PWM block changed: add one data register PWMDATA1 • Operating voltage range change to 1.8V-5.5V	• For improve the compatibility without conflict and confusion • Spec change to fit the real performance.
	Dec. 23, 2008	• Internal RC precision change to : Typ. 3%, max: 5% • ADC maximum clock = 2MHz • ADC operating voltage = 2.7V - 5.5V	• For more margin to design. vAccording to the minimum convert time.
V0.2	Jan. 07, 2009	• Interrupt low width : 200nS (typ.) --same with 9454 • Reset width : 1uS(Min.) --same with 9454 • LVR current added → IDD3 (stop with LVR enable) data added. • Reason: – 2.6V,250C,LVR enable parameter same with 9454: 30uA (typ.) 60uA (Max.) (simulation result: 22uA typ.) (09-06 test chip test result: 28uA @ 2.6V) – 5V, 250C,LVR enable: simulation result: 35uA, so according to the relation of current with voltage, set the parameter as 50uA(Typ.) 100uA(Max.) （5V-2.6V, 30uA – 50uA）.	• According to new noise filter design. • For the reason of operating voltage change.

Revision No.	Date	Description	Remark
		(9454 parameter too large) (09-06 test chip test result: 38uA @ 5.0V) • LVR Level: add 1.9V	
V0.3	Feb. 09, 2009	No	No
	Mar. 02, 2009	• Package: delete 16SSOP(not available in Samsung) • Package: change 16SOP-BD300-SG to 16SOP-225 (16SOP-BD300 not available in Samsung) • Package: add 16-TSSOP-BD44.	• Some Package types not available any more.
	Mar. 11, 2009	• ADC maximum clock: 4MHz • ADC conversion time: Minimum: 12.5us. • ADC operating voltage range: 2.0V-5.5V	• According to LF13H ADC test result
V0.4	Mar. 13, 2009	• Product: S3F94C4 → S3F94C8/F94C4 • Add C8 related contents. • CPU core type: SAM8RCRI → SAM88RCRI • VPP description change: voltage for tool mode entering. • Smart option : add 1.9V level(10100) • Address Space: C8-8K. • 16TSSOP Package type confirmed: 16TSSOP-0044	
	Apr. 15, 2009	• Change description : internal RC, VDD = 5 V, TA =25°C, Tolerance: 1%	• Emphasis the precision of 1%
	Apr. 17, 2009	• LVR level： 2.3V: Change error: Min：2.0, Typ:2.3, Max: 2.6	• Error correction.
V0.5	Mar. 25, 2009	• IDD3 change : Typical vale @ 25C,4.5V-5.5V change from 1.0uA to 0.3uA • PWM Spec : – Add figure for PWMEX and PWM data register description – Change PWM block diagram according to real circuit: PWM extension bit no buffer, so delete it in the diagram.	• According S3F80PB test result and PE's suggestion. • According 9454 real circuit and test result • For describe the PWM module more clearly.
V0.6	Jul. 30, 2009	• Add description about Information Block in Chap.16 • Chap.17 modified according to new Target board layout. • Delete 16-pin DIP package type. (Chap 1/15) • Change minimum operating frequency to 0.4MHz • Before: 1-10MHz; Now: 0.4-10MHz • Add contents about external reset circuit. • Add Notes about the unused bits of FLAGs register	• New feature, has been verified • Marketing required.

Revision No.	Date	Description	Remark
		• ADC example contents refine. • Electrical parameter: ADC operating current @3.0V confirmed—Typ.0.15mA, max 0.45mA	
		• Change Flash Endurance to 100,000 times. (YFH/CHT) – Original: 10,000 times.	
V0.7	Sep. 29, 2009	• P1.2 Pull-down resistor related contents changed/ added – Electrical data: ILIH1/ILIH2 changed ILIH1: condition–All input ports except ILIH2 and P1.2 ILIH2: Xin (delete Xout, because it's output) ILIL2: Xin (delete Xout, because it's output) And note below the table to describe the pull-down resistor of P1.2 – Add note in chapter 09, notice customer to pay attention to the pull-down resistor of P1.2 – Add note in chapter 01, notice that P1.2 have intrinsic pull-down resistor.	• P1.2 is VPP pin, it has intrinsic internal pull-down resistor about 300Kohm. • It's different with 9454 • The pull-down resistor has been changed from 100Kohm to 300Kohm in Main chip • Spec will be updated and confirmed after test main chip.
	Oct.19, 2009	• Change Flash Endurance to 10,000 times. (CHT) – Original: 100,000 times. – Reason: According to Mr. Chung's information: – 250°C : 100,000 – -40 - +85: 10,000 – So, Spec: 10,000	• Chapter 1/13/14
	Nov. 24, 2009	• Add Note for PWMCON.5: should always be set to '0' Because EVA chip not revision. So need to note in Spec.	• Chapter 12
	Dec. 01, 2009	• PWM extended waveform of 6+6 and 8+6 modified: – Correct error: PWMEX value in the figure	• Chapter 11. Error correction.

Preface

The S3F94C8/94C4 Microcontroller User's Manual is designed for application designers and programmers who are using the S3F94C8/94C4 microcontroller for application development. It is organized in two main parts:

Part I Programming Model

Part II Hardware Descriptions

Part I contains software-related information to familiarize you with the microcontroller's architecture, programming model, instruction set, and interrupt structure. It has six chapters:

Chapter 1	Product Overview	Chapter 4	Control Registers
Chapter 2	Address Spaces	Chapter 5	Interrupt Structure
Chapter 3	Addressing Modes	Chapter 6	Instruction Set

Chapter 1, "Product Overview," is a high-level introduction to S3F94C8/94C4 with general product descriptions, as well as detailed information about individual pin characteristics and pin circuit types.

Chapter 2, "Address Spaces," describes program and data memory spaces, the internal register file, and register addressing. Chapter 2 also describes working register addressing, as well as system stack and user-defined stack operations.

Chapter 3, "Addressing Modes," contains detailed descriptions of the addressing modes that are supported by the S3F8-series CPU.

Chapter 4, "Control Registers," contains overview tables for all mapped system and peripheral control register values, as well as detailed one-page descriptions in a standardized format. You can use these easy-to-read, alphabetically organized, register descriptions as a quick-reference source when writing programs.

Chapter 5, "Interrupt Structure," describes the S3F94C8/94C4 interrupt structure in detail and further prepares you for additional information presented in the individual hardware module descriptions in Part II.

Chapter 6, "Instruction Set," describes the features and conventions of the instruction set used for all S3F8-series microcontrollers. Several summary tables are presented for orientation and reference. Detailed descriptions of each instruction are presented in a standard format. Each instruction description includes one or more practical examples of how to use the instruction when writing an application program.

A basic familiarity with the information in Part I will help you to understand the hardware module descriptions in Part II. If you are not yet familiar with the S3F8-series microcontroller family and are reading this manual for the first time, we recommend that you first read Chapters 1-3 carefully. Then, briefly look over the detailed information in Chapters 4, 5, and 6. Later, you can reference the information in Part I as necessary.

Part II "hardware Descriptions," has detailed information about specific hardware components of the S3F94C8/94C4 microcontroller. Also included in Part II are electrical, mechanical, flash, and development tools data. It has 9 chapters:

Chapter 7	Clock Circuit	Chapter 13	Embedded Flash Interface
Chapter 8	RESET and Power-Down	Chapter 14	Electrical Data
Chapter 9	I/O Ports	Chapter 15	Mechanical Data
Chapter 10	Basic Timer and Timer 0	Chapter 16	Flash MCU
Chapter 11	12-bit PWM	Chapter 17	Development Tools
Chapter 12	A/D Converter		

Table of Contents

1 OVERVIEW OF S3F94C8/94C4 1-1

1.1 SAM88RCRI Microcontrollers 1-1
1.1.1 S3F94C8/F94C4 Microcontroller 1-1
1.2 Features 1-2
1.3 Block Diagram of S3F94C8/94C8 1-3
1.4 Pin Assignments 1-4
1.5 Pin Descriptions 1-6
1.6 Pin Circuits 1-8

2 ADDRESS SPACES 2-1

2.1 Overview of Address Spaces 2-1
2.2 Program Memory (ROM) 2-2
2.2.1 Normal Operating Mode 2-2
2.2.2 Smart Option 2-3
2.3 Register Architecture 2-5
2.4 Common Working Register Area (C0H–CFH) 2-7
2.5 System Stack 2-8
2.5.1 Stack Operations 2-8
2.5.2 Stack Pointer (SP) 2-8

3 ADDRESSING MODES 3-1

3.1 Overview of Addressing Modes 3-1
3.1.1 Register Addressing Mode 3-2
3.1.2 Indirect Register Addressing Mode 3-3
3.1.3 Indirect Register Addressing Mode (1) 3-4
3.1.4 Indirect Register Addressing Mode (2) 3-5
3.1.5 Indirect Register Addressing Mode (3) 3-6
3.1.6 Indexed Addressing Mode 3-7
3.1.7 Indexed Addressing Mode (1) 3-8
3.1.8 Indexed Addressing Mode (2) 3-9
3.1.9 Direct Address Mode 3-10
3.1.10 Direct Address Mode (1) 3-11
3.1.11 Relative Address Mode 3-12
3.1.12 Immediate Mode 3-12

4 ADDRESSING REGISTERS 4-1

4.1 Overview of Control Registers 4-1
4.1.1 ADCON — A/D Converter Control Register: F7H 4-5
4.1.2 BTCON — Basic Timer Control Register: DCH 4-6
4.1.3 CLKCON — Clock Control Register: D4H 4-7
4.1.4 FLAGS — System Flags Register: D5H 4-8
4.1.5 FMCON — Flash Memory Control Register: ECH 4-9
4.1.6 FMSECH — Flash Memory Sector Address Register (High Byte): EEH 4-10
4.1.7 FMSECL — Flash Memory Sector Address Register (Low Byte): EFH 4-10
4.1.8 FMUSR — Flash Memory User Programming Enable Register: EDH 4-10
4.1.9 P0CONH — Port 0 Control Register (High Byte): E6H 4-11

4.1.10 P0CONL — Port 0 Control Register (Low Byte): E7H 4-12
4.1.11 P0PND — Port 0 Interrupt Pending Register: E8H 4-13
4.1.12 P1CON — Port 1 Control Register: E9H 4-14
4.1.13 P2CONH — Port 2 Control Register (High Byte): EAH 4-15
4.1.14 P2CONL — Port 2 Control Register (Low Byte): EBH 4-16
4.1.15 PWMCON — PWM Control Register: F3H 4-17
4.1.16 PWMEX — PWM Extension Register: F1H 4-18
4.1.17 STOPCON — STOP Mode Control Register: E4H 4-19
4.1.18 SYM — System Mode Register: DFH 4-19
4.1.19 T0CON — TIMER 0 Control Register: D2H 4-20

5 INTERRUPT STRUCTURE 5-1

5.1 Overview of Interrupt Structure 5-1
5.1.1 Interrupt Processing Control Points 5-1
5.1.2 Enable/Disable Interrupt Instructions (EI, DI) 5-2
5.1.3 Interrupt Pending Function Types 5-2
5.1.4 Interrupt Priority 5-2
5.1.5 Interrupt Source Service Sequence 5-3
5.1.6 Interrupt Service Routines 5-3
5.1.7 Generating interrupt Vector Addresses 5-3
5.1.8 S3F94C8/F94C4 Interrupt Structure 5-4
5.1.9 Peripheral Interrupt Control Registers 5-5

6 SAM88RCRI INSTRUCTION SET 6-1

6.1 Overview of SAM88RCRI Instruction Set 6-1
6.1.1 Register Addressing 6-1
6.1.2 Addressing Modes 6-1
6.1.3 Flags Register (FLAGS) 6-4
6.1.4 Flag Descriptions 6-4
6.1.4.1 Overflow Flag (FLAGS.4, V) 6-4
6.1.4.2 Sign Flag (FLAGS.5, S) 6-4
6.1.4.3 Zero Flag (FLAGS.6, Z) 6-4
6.1.4.4 Carry Flag (FLAGS.7, C) 6-4
6.1.5 Instruction Set Notation 6-5
6.1.6 Condition Codes 6-9
6.1.7 Instruction Descriptions 6-10
6.1.7.1 ADC — Add with Carry 6-11
6.1.7.2 ADD — Add 6-12
6.1.7.3 AND — Logical AND 6-13
6.1.7.4 CALL — Call Procedure 6-14
6.1.7.5 CCF — Complement Carry Flag 6-15
6.1.7.6 CLR — Clear 6-16
6.1.7.7 COM — Complement 6-17
6.1.7.8 CP — Compare 6-18
6.1.7.9 DEC — Decrement 6-19
6.1.7.10 DI — Disable Interrupts 6-20
6.1.7.11 EI — Enable Interrupts 6-21
6.1.7.12 IDLE — Idle Operation 6-22
6.1.7.13 INC — Increment 6-23
6.1.7.14 IRET — Interrupt Return 6-24
6.1.7.15 JP — Jump 6-25
6.1.7.16 JR — Jump Relative 6-26

6.1.7.17 LD — Load 6-27
6.1.7.18 LDC/LDE — Load Memory 6-29
6.1.7.19 LDCD/LDED — Load Memory and Decrement 6-32
6.1.7.20 LDCI/LDEI — Load Memory and Increment 6-33
6.1.7.21 NOP — No Operation 6-34
6.1.7.22 OR — Logical OR 6-35
6.1.7.23 POP — Pop From Stack 6-36
6.1.7.24 PUSH — Push To Stack 6-37
6.1.7.25 RCF — Reset Carry Flag 6-38
6.1.7.26 RET — Return 6-39
6.1.7.27 RL — Rotate Left 6-40
6.1.7.28 RLC — Rotate Left Through Carry 6-41
6.1.7.29 RR — Rotate Right 6-42
6.1.7.30 RRC — Rotate Right Through Carry 6-43
6.1.7.31 SBC — Subtract With Carry 6-44
6.1.7.32 SCF — Set Carry Flag 6-45
6.1.7.33 SRA — Shift Right Arithmetic 6-46
6.1.7.34 STOP — Stop Operation 6-47
6.1.7.35 SUB — Subtract 6-48
6.1.7.36 TCM — Test Complement Under Mask 6-49
6.1.7.37 TM — Test Under Mask 6-50
6.1.7.38 XOR — Logical Exclusive OR 6-51

7 CLOCK CIRCUIT 7-1

7.1 Overview of Clock Circuit 7-1
7.1.1 Main Oscillator Logic 7-2
7.1.2 Clock Status During Power-Down Modes 7-2
7.1.3 System Clock Control Register (CLKCON) 7-2

8 RESET AND POWER-DOWN 8-1

8.1 System Reset 8-1
8.1.1 Overview of System Reset 8-1
8.1.1.1 External Reset Pin 8-3
8.1.1.2 MCU Initialization Sequence 8-4
8.2 Power-Down Modes 8-5
8.2.1 Stop Mode 8-5
8.2.1.1 Using RESET to Release Stop Mode 8-5
8.2.1.2 Using an External Interrupt to Release Stop Mode 8-5
8.2.2 Idle Mode 8-6
8.3 Hardware Reset Values 8-7

9 I/O PORTS 9-1

9.1 Overview of I/O Ports 9-1
9.1.1 Port Data Registers 9-2
9.1.2 Port 0 9-3
9.1.3 Port 1 9-7
9.1.4 Port 2 9-9

10 BASIC TIMER AND TIMER 0 10-1

10.1 Overview of Basic Timer and Timer 0 10-1
10.1.1 Basic Timer 10-1

10.1.2 Timer 0 10-1
10.2 Basic Timer (BT) 10-2
10.2.1 Basic Timer Control Register (BTCON) 10-2
10.2.2 Basic Timer Function 10-3
10.2.2.1 Watchdog Timer Function 10-3
10.2.2.2 Oscillation Stabilization Interval Timer Function 10-3
10.3 Timer 0 10-7
10.3.1 Timer 0 Control Registers (T0CON) 10-7
10.3.2 Timer 0 Function Description 10-8
10.3.2.1 Interval Timer Mode 10-8

11 PULSE WIDTH MODULATION 11-1

11.1 Overview of Pulse Width Modulation 11-1
11.1.1 Function Description 11-1
11.1.1.1 PWM 11-1
11.1.1.2 PWM Counter 11-1
11.1.1.3 PWM Data and Extension Registers 11-2
11.1.1.4 PWM Clock Rate 11-3
11.1.1.5 PWM Function Description 11-3
11.1.1.6 PWM Output Waveform 11-4
11.1.2 PWM Control Register (PWMCON) 11-10
11.1.3 PWM Extension Register (PWMEX) 11-11

12 A/D CONVERTER 12-1

12.1 Overview of A/D Converter 12-1
12.1.1 Using A/D Pins for Standard Digital Input 12-2
12.1.2 A/D Converter Control Register (ADCON) 12-2
12.1.3 Internal Reference Voltage Levels 12-3
12.1.4 Conversion Timing 12-4
12.1.5 Internal A/D Conversion Procedure 12-5

13 EMBEDDED FLASH MEMORY INTERFACE 13-1

13.1 Overview of Embedded Flash Memory Interface 13-1
13.1.1 Flash ROM Configuration 13-1
13.1.1.1 Tool Program Mode 13-2
13.1.1.2 User Program Mode 13-2
13.2 Flash Memory Control Registers (User Program Mode) 13-3
13.2.1 Flash Memory Control Register (FMCOn) 13-3
13.2.2 Flash Memory User Programming Enable Register (FMUSR) 13-3
13.2.3 Flash Memory Sector Address Registers 13-4
13.3 Sector Erase 13-5
13.3.1 The Sector Erase Procedure in User Program Mode 13-6
13.4 Programming 13-8
13.4.1 Program Procedure in User Program Mode 13-8
13.5 Reading 13-14
13.5.1 Program Procedure in User Program Mode 13-14
13.6 Hard Lock Protection 13-15
13.6.1 Program Procedure in User Program Mode 13-15

14 ELECTRICAL DATA 14-1

14.1 Overview of Electrical Data 14-1

15 MECHANICAL DATA 15-1
15.1 Overview of Mechanical Data 15-1
16 S3F94C8/F94C4 FLASH MCU 16-1
16.1 Overview of Flash MCU 16-1
16.2 On Board Writing 16-4
16.2.1 Circuit Design Guide 16-4
16.3 Information Block 16-6
17 DEVELOPMENT TOOLS 17-1
17.1 Overview of Development Tools 17-1
17.1.1 Target Boards 17-1
17.1.2 Programming Socket Adapter 17-1
17.1.3 Development System Configuration 17-2
17.1.4 TB94C8/94C4 Target Board 17-3
17.1.5 SMDS2+ Selection (SAM8) 17-5
17.1.6 Third parties for Development Tools 17-9
17.1.6.1 In-Circuit Emulator for SAM8 family 17-9
17.1.6.2 OTP/MTP Programmer 17-9
17.1.6.3 Development Tools Suppliers 17-9
17.1.6.4 8-bit In-Circuit Emulator 17-10

List of Figures

Figure Number	Title	Page Number
Figure 1-1	Block Diagram of S3F94C8/94C8	1-3
Figure 1-2	Pin Assignment Diagram (20-Pin DIP/SOP/SSOP Package)	1-4
Figure 1-3	Pin Assignment Diagram (16-Pin SOP/TSSOP Package)	1-5
Figure 1-4	Pin Circuit Type A	1-8
Figure 1-5	Pin Circuit Type B (P1.2)	1-8
Figure 1-6	Pin Circuit Type C	1-8
Figure 1-7	Pin Circuit Type D	1-9
Figure 1-8	Pin Circuit Type E (Port2)	1-9
Figure 1-9	Pin Circuit Type E-1 (Port0)	1-10
Figure 1-10	Pin Circuit Type E-2 (P1.0-P1.1)	1-10
Figure 2-1	Program Memory Address Space	2-2
Figure 2-2	Smart Option	2-3
Figure 2-3	Internal Register File Organization	2-6
Figure 2-4	16-Bit Register Pairs	2-7
Figure 2-5	Stack Operations	2-8
Figure 3-1	Register Addressing	3-2
Figure 3-2	Working Register Addressing	3-2
Figure 3-3	Indirect Register Addressing to Register File	3-3
Figure 3-4	Indirect Register Addressing to Program Memory	3-4
Figure 3-5	Indirect Working Register Addressing to Register File	3-5
Figure 3-6	Indirect Working Register Addressing to Program or Data Memory	3-6
Figure 3-7	Indexed Addressing to Register File	3-7
Figure 3-8	Indexed Addressing to Program or Data Memory with Short Offset	3-8
Figure 3-9	Indexed Addressing to Program or Data Memory with Long Offset	3-9
Figure 3-10	Direct Addressing for Load Instructions	3-10
Figure 3-11	Direct Addressing for Call and Jump Instructions	3-11
Figure 3-12	Relative Addressing	3-12
Figure 3-13	Immediate Addressing	3-12
Figure 4-1	Register Description Format	4-4
Figure 5-1	S3F9-Series Interrupt Type	5-1
Figure 5-2	Interrupt Function Diagram	5-2
Figure 5-3	S3F94C8/F94C4 Interrupt Structure	5-4
Figure 6-1	System Flags Register (FLAGS)	6-4
Figure 7-1	Main Oscillator Circuit (RC Oscillator with Internal Capacitor)	7-1
Figure 7-2	Main Oscillator Circuit (Crystal/Ceramic Oscillator)	7-1
Figure 7-3	System Clock Control Register (CLKCON)	7-2
Figure 7-4	System Clock Circuit Diagram	7-3

Figure 8-1 Low Voltage Reset Circuit 8-2
Figure 8-2 Recommended External RESET Circuit 8-3
Figure 8-3 Reset Block Diagram 8-4
Figure 8-4 Timing for S3F94C8/F94C4 After RESET 8-4

Figure 9-1 Port Data Register Format 9-2
Figure 9-2 Port 0 Circuit Diagram 9-3
Figure 9-3 Port 0 Control Register (P0CONH, High Byte) 9-4
Figure 9-4 Port 0 Control Register (P0CONL, Low Byte) 9-5
Figure 9-5 Port 0 Interrupt Pending Registers (P0PND) 9-6
Figure 9-6 Port 1 Circuit Diagram 9-7
Figure 9-7 Port 1 Control Register (P1CON) 9-8
Figure 9-8 Port 2 Circuit Diagram 9-9
Figure 9-9 Port 2 Control Register (P2CONH, High Byte) 9-10
Figure 9-10 Port 2 Control Register (P2CONL, Low Byte) 9-1

Figure 10-1 Basic Timer Control Register (BTCON) 10-2
Figure 10-2 Oscillation Stabilization Time on RESET 10-4
Figure 10-3 Oscillation Stabilization Time on STOP Mode Release 10-5
Figure 10-4 Timer 0 Control Registers (T0CON) 10-7
Figure 10-5 Simplified Timer 0 Function Diagram (Interval Timer Mode) 10-8
Figure 10-6 Timer 0 Timing Diagram 10-9
Figure 10-7 Basic Timer and Timer 0 Block Diagram 10-10

Figure 11-1 PWM Data and Extension Registers 11-2
Figure 11-2 PWM Basic Waveform (6-bit base) 11-4
Figure 11-3 Extended PWM Waveform (6-bit base + 2-bit extension) 11-5
Figure 11-4 PWM Basic Waveform (6-bit base) 11-6
Figure 11-5 Extended PWM Waveform (6-bit base + 6-bit extension) 11-7
Figure 11-6 PWM Basic Waveform (8-bit base) 11-8
Figure 11-7 PWM Basic Waveform (8-bit base + 6-bit extension) 11-9
Figure 11-8 PWM Control Register (PWMCON) 11-10
Figure 11-9 PWM Extension Register (PWMEX) 11-11
Figure 11-10 PWM Data Register (PWMDATA) 11-11
Figure 11-11 PWM Module Functional Block Diagram 11-12

Figure 12-1 A/D Converter Control Register (ADCON) 12-2
Figure 12-2 A/D Converter Circuit Diagram 12-3
Figure 12-3 A/D Converter Data Register (ADDATAH/L) 12-3
Figure 12-4 A/D Converter Timing Diagram 12-4
Figure 12-5 Recommended A/D Converter Circuit for Highest Absolute Accuracy 12-5

Figure 13-1 Flash Memory Control Register (FMCON) 13-3
Figure 13-2 Flash Memory User Programming Enable Register (FMUSR) 13-3
Figure 13-3 Flash Memory Sector Address Register (FMSECH) 13-4
Figure 13-4 Flash Memory Sector Address Register (FMSECL) 13-4
Figure 13-5 Sector configurations in User Program Mode 13-5
Figure 13-6 Sector Erase Flowchart in User Program Mode 13-6
Figure 13-7 Byte Program Flowchart in a User Program Mode 13-9
Figure 13-8 Program Flowchart in a User Program Mode 13-10

Figure 14-1 Input Timing Measurement Points 14-4
Figure 14-2 Operating Voltage Range 14-7
Figure 14-3 Schmitt Trigger Input Characteristics Diagram 14-7
Figure 14-4 Stop Mode Release Timing When Initiated by a RESET 14-8
Figure 14-5 LVR Reset Timing 14-9
Figure 14-6 Circuit Diagram to Improve EFT Characteristics 14-10

Figure 15-1 20-DIP-300A Package Dimensions 15-1
Figure 15-2 20-SOP-375 Package Dimensions 15-2
Figure 15-3 20-SSOP-225 Package Dimensions 15-3
Figure 15-4 16-SOP-225 Package Dimensions 15-4
Figure 15-5 16-TSSOP-0044 Package Dimensions 15-5

Figure 16-1 S3F94C8/F94C4 Pin Assignments (20-DIP/20SOP) 16-2
Figure 16-2 S3F94C8/F94C4 Pin Assignments (16SOP) 16-2
Figure 16-3 PCB design guide for on-board programming 16-4
Figure 16-4 S3F94C8/F94C4 Flash Architecture 16-6

Figure 17-1 Development System Configuration 17-2
Figure 17-2 TB94C8/94C4 Target Board Configuration 17-3
Figure 17-3 DIP Switch for Smart Option 17-7
Figure 17-4 20-Pin Connector for TB94C8/94C4 17-8
Figure 17-5 S3F94C8/F94C4 Probe Adapter for 20-DIP Package 17-8

List of Tables

Table Number	Title	Page Number
Table 1-1	S3F94C8/F94C4 Pin Descriptions	1-6
Table 1-2	Descriptions of Pins Used to Read/Write the Flash ROM	1-7
Table 2-1	Register Type Summary	2-5
Table 4-1	System and Peripheral Control Registers	4-2
Table 5-1	Interrupt Source Control and Data Registers	5-5
Table 6-1	Instruction Group Summary	6-2
Table 6-2	Flag Notation Conventions	6-5
Table 6-3	Instruction Set Symbols	6-5
Table 6-4	Instruction Notation Conventions	6-6
Table 6-5	Opcode Quick Reference	6-7
Table 6-6	Opcode Quick Reference (1)	6-8
Table 6-7	Condition Codes	6-9
Table 8-1	Register Values after a Reset	8-7
Table 9-1	Overview of S3F94C8/F94C4 Port Configuration	9-1
Table 9-2	Port Data Register Summary	9-2
Table 11-1	PWM Control and Data Registers	11-3
Table 11-2	PWM output “stretch” Values for Extension Data bits Ext0 (PWMDATA.1–.0)	11-4
Table 11-3	PWM output “stretch” Values for Extension Data bits Ext1 (PWMEX.7-.2)	11-6
Table 11-4	PWM output “stretch” Values for Extension Data bits Ext1 (PWMEX.7-.2)	11-8
Table 13-1	Descriptions of Pins Used to Read/Write the Flash in Tool Program Mode	13-2
Table 14-1	Absolute Maximum Ratings	14-2
Table 14-2	DC Electrical Characteristics	14-3
Table 14-3	AC Electrical Characteristics	14-4
Table 14-4	Oscillator Characteristics	14-5
Table 14-5	Oscillation Stabilization Time	14-5
Table 14-6	RC Oscillator Characteristics (S3F94C8EZZ / F94C4EZZ)	14-6
Table 14-7	RC Oscillator Characteristics (S3F94C8XZZ / F94C4XZZ)	14-6
Table 14-8	Data Retention Supply Voltage in Stop Mode	14-7
Table 14-9	A/D Converter Electrical Characteristics	14-8
Table 14-10	LVR Circuit Characteristics	14-9
Table 14-11	Flash Memory AC Electrical characteristics	14-9
Table 14-12	ESD Characteristics	14-10

Table 16-1 Descriptions of Pins Used to Read/Write the EPROM 16-3
Table 16-2 Comparison of S3F94C8/F94C4 Features 16-3
Table 16-3 Reference Table for Connection 16-5
Table 16-4 Operation Results Comparison of Main Flash ROM and Information Blocks 16-6

Table 17-1 Components of TB94C8/94C4 17-4
Table 17-2 Power Selection Settings for TB94C8/94C4 17-4
Table 17-3 The SMDS2+ Tool Selection Setting 17-5
Table 17-4 Using Single Header Pins to Select Clock Source / PWM / Operation Mode 17-6
Table 17-5 Using Single Header Pins as the Input Path for External Trigger Sources 17-7

List of Examples

Example Number	Title	Page Number
Example 2-1	Programming Tip — Smart Option Setting	2-4
Example 2-2	Programming Tip — Addressing the Common Working Register Area	2-7
Example 2-3	Programming Tip — Standard Stack Operations Using PUSH and POP	2-9
Example 8-1	Programming Tip — Sample S3F94C8/F94C4 Initialization Routine	8-9
Example 10-1	Programming Tip — Configuring the Basic Timer	10-6
Example 10-2	Programming Tip — Configuring Timer 0 (Interval Mode)	10-11
Example 11-1	Programming Tip — Programming the PWM Module to Sample Specifications	11-13
Example 12-1	Programming Tip — Configuring A/D Converter	12-6
Example 13-1	Programming Tip — Sector Erase	13-7
Example 13-2	Programming Tip — Programming	13-11
Example 13-3	Programming Tip — Reading	13-14
Example 13-4	Programming Tip — Hard Lock Protection	13-15

1 OVERVIEW OF S3F94C8/94C4

1.1 SAM88RCRI Microcontrollers

Samsung's SAM88RCRI series of 8-bit single-chip CMOS microcontrollers offer a fast and efficient CPU along with a wide range of integrated peripherals and programmable ROM with various sizes.

The key features of CPU include:

- Efficient register-oriented architecture
- Selectable CPU clock sources
- Idle and Stop power-down modes released by the interrupt
- Built-in basic timer with watchdog function

To render a flexible programming environment for applications with varied memory and I/O requirements, an address/data bus architecture and a large number of bit-configurable I/O ports have been provided.

To support real-time operation(s), timer/counters with selectable operating modes are also included.

1.1.1 S3F94C8/F94C4 Microcontroller

The S3F94C8/F94C4 single-chip 8-bit CMOS microcontroller is designed for A/D converter application field. This microcontroller, which is based on Samsung's SAM88RCRI CPU architecture, is fabricated using an advanced CMOS process. To reduce power consumption, Stop and Idle (Power-down) modes are implemented.

The S3F94C8 microcontroller contains an embedded 8Kbyte multi-time programmable Full Flash ROM. On the other hand, the S3F94C4 microcontroller contains an 4Kbyte multi-time programmable Full Flash ROM.

Both S3C94C8 and S3F94C4 are general-purpose microcontrollers that are ideal for use in a wide range of electronics applications requiring simple timer/counter and Pulse-width Modulation (PWM). In addition, the advanced CMOS technology in S3F94C8/F94C4 accounts for low power consumption and wide operating voltage range.

Using the SAM88RCRI design approach, the following peripherals are integrated with the SAM8RCRI core:

- Three configurable I/O ports (18 pins)
- Four interrupt sources with one vector and one interrupt level
- 0ne 8-bit timer/counter with time interval modes
- Analog-to-digital converter with nine input channels (maximum) and 10-bit resolution
- One PWM output with three optional modes: 8-bit (6+2), 12-bit (6+6), and 14-bit (8+6)

The S3F94C8/F94C4 microcontroller is ideal for use in a wide range of electronic applications requiring simple timer/counter, PWM, and ADC. They are currently available in 20 DIP Package, 20-/16-pin SOP Package, 20 SSOP Package and 16 TSSOP Package.

1.2 Features

CPU
- SAM88RCRI CPU core

Memory
- Internal multi-time program Full-Flash memory:
  - 8K×8 bits program memory(S3F94C8)
  - 4K×8 bits program memory(S3F94C4)
    - Sector size: 128 Bytes
    - User programmable by 'LDC' instruction
    - Sector erase available
    - Fast programming time
    - External serial programming support
    - Endurance: 10,000 erase/program cycles
    - 10 Years data retention
- 208-byte general-purpose register area

Instruction Set
- 41 instructions
- Idle and Stop instructions added for power-down modes

Instruction Execution Time
- 400 ns at 10 MHz fOSC (minimum)

Interrupts
- 1 interrupt levels and 4 interrupt sources (2 external interrupts and 2 internal interrupts)

General I/O
- Three I/O ports (Max 18 pins)
- Bit programmable ports

1-ch High-speed PWM with Three Selectable Resolutions
- 8-bit PWM: 6-bit base + 2-bit extension
- 12-bit PWM: 6-bit base + 6-bit extension
- 14-bit PWM: 8-bit base + 6-bit extension

Timer/Counters
- One 8-bit basic timer for watchdog function
- One 8-bit timer/counter with time interval modes

A/D Converter
- Nine analog input pins (MAX)
- 10-bit conversion resolution

Oscillation Frequency
- 0.4 MHz to 10 MHz external crystal oscillator
- Typical 4MHz external RC oscillator
- Internal RC: 3.2 MHz (typ.), 0.5 MHz (typ.) in VDD = 5 V

Built-in RESET Circuit (LVR)
- Low-Voltage check to make system reset
- V_{LVR} = 1.9/2.3/3.0/3.6/3.9 V (by smart option)

Smart Option
- LVR enable/disable
- Oscillator selection

Operating Temperature Range
- – 40°C to + 85°C

Operating Voltage Range
- 1.8 V to 5.5 V @ 0.4 - 4M Hz(LVR disable)
- LVR to 5.5V @ 0.4 - 4M Hz(LVR enable)
- 2.7 V to 5.5V @ 0.4 -10M Hz

Package Types
- S3F94C8/F94C4:
  - 20-DIP-300A
  - 20-SOP-375
- 20-SSOP-225
  - 16-SOP-225
- 16-TSSOP-0044

Device	Operating Temp. Range	Internal RC Temp. Range	Internal RC Tolerance
S3F94C8EZZ / F94C4EZZ	– 40°C to + 85°C	– 25°C to + 85°C	3%@5V,25°C
S3F94C8XZZ / F94C4XZZ	– 40°C to + 85°C	– 40°C to + 85°C	1%@5V,25°C

1.3 Block Diagram of S3F94C8/94C8

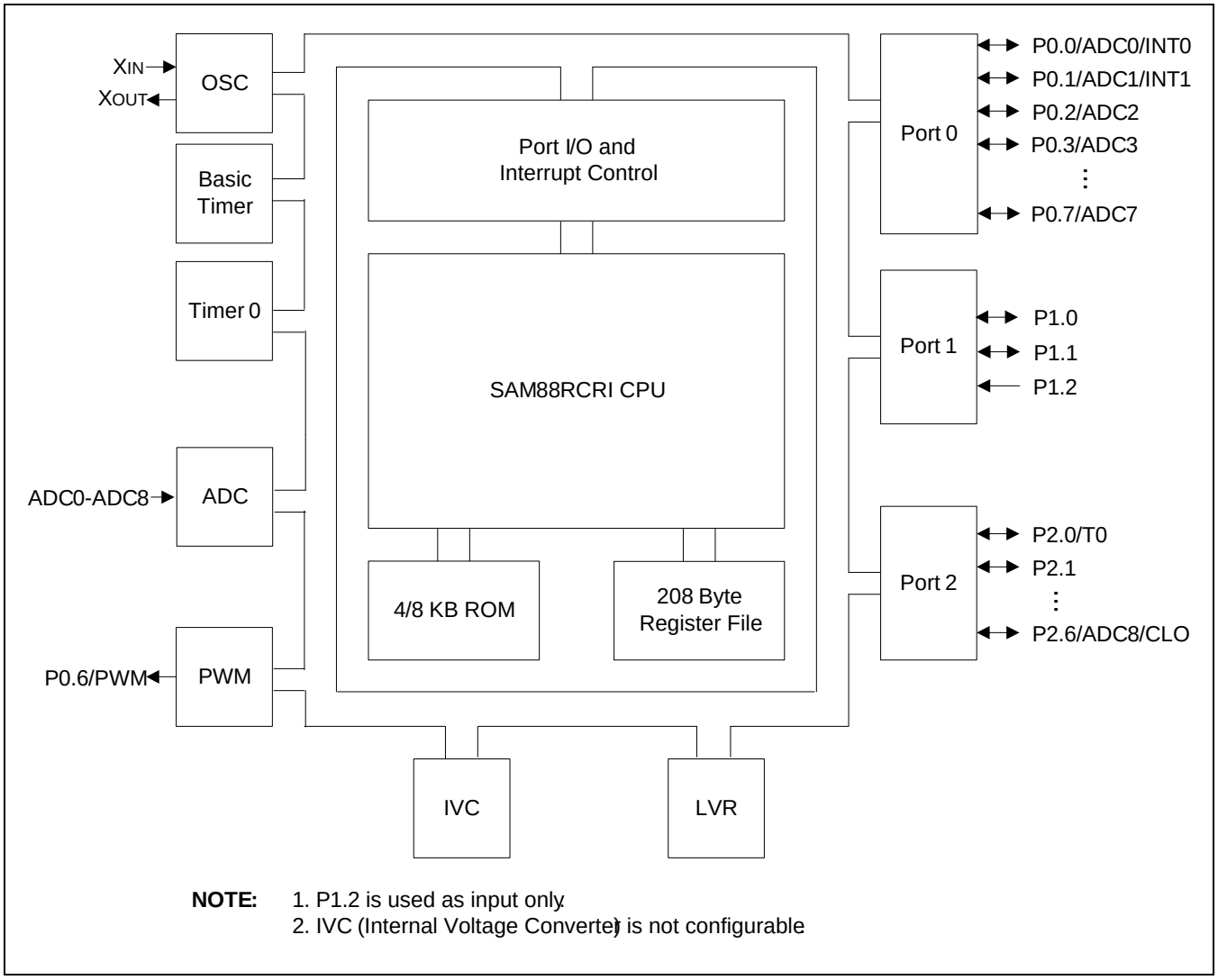


Figure 1-1 Block Diagram of S3F94C8/94C8

1.4 Pin Assignments

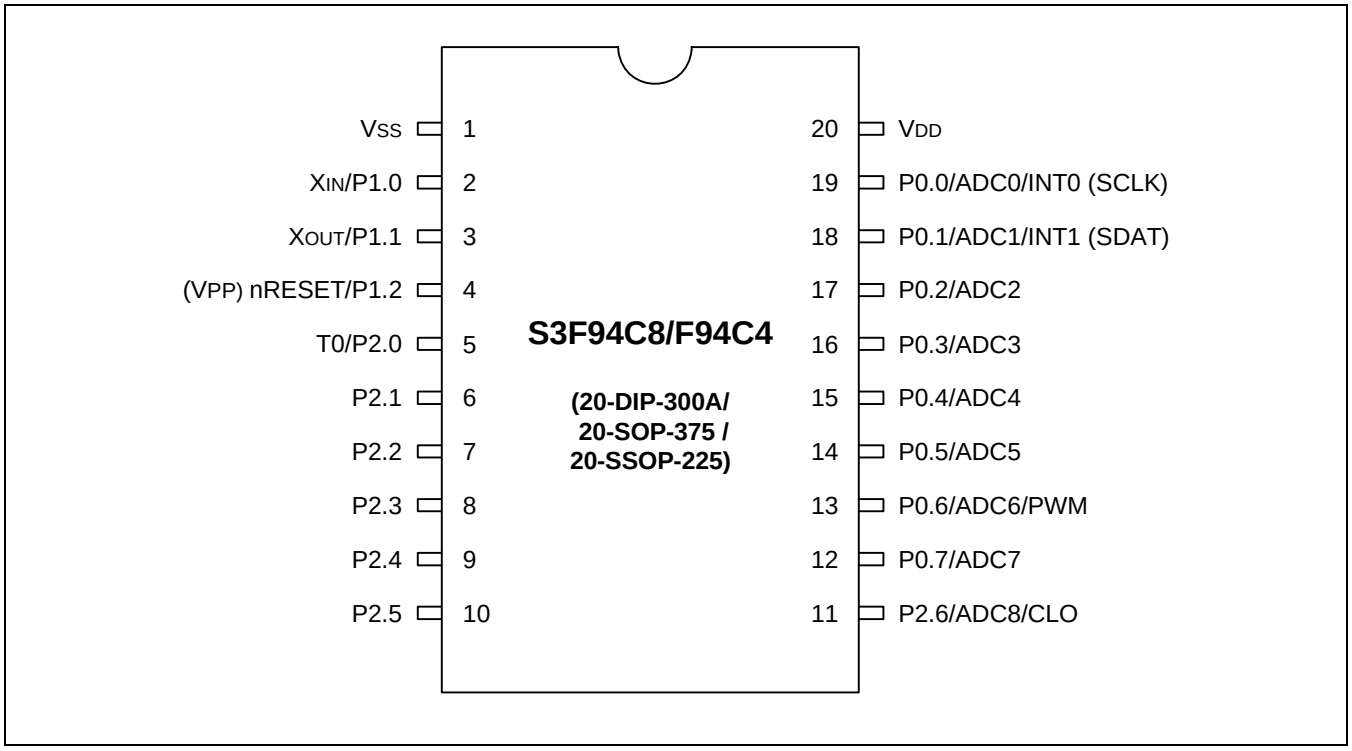


Figure 1-2 Pin Assignment Diagram (20-Pin DIP/SOP/SSOP Package)

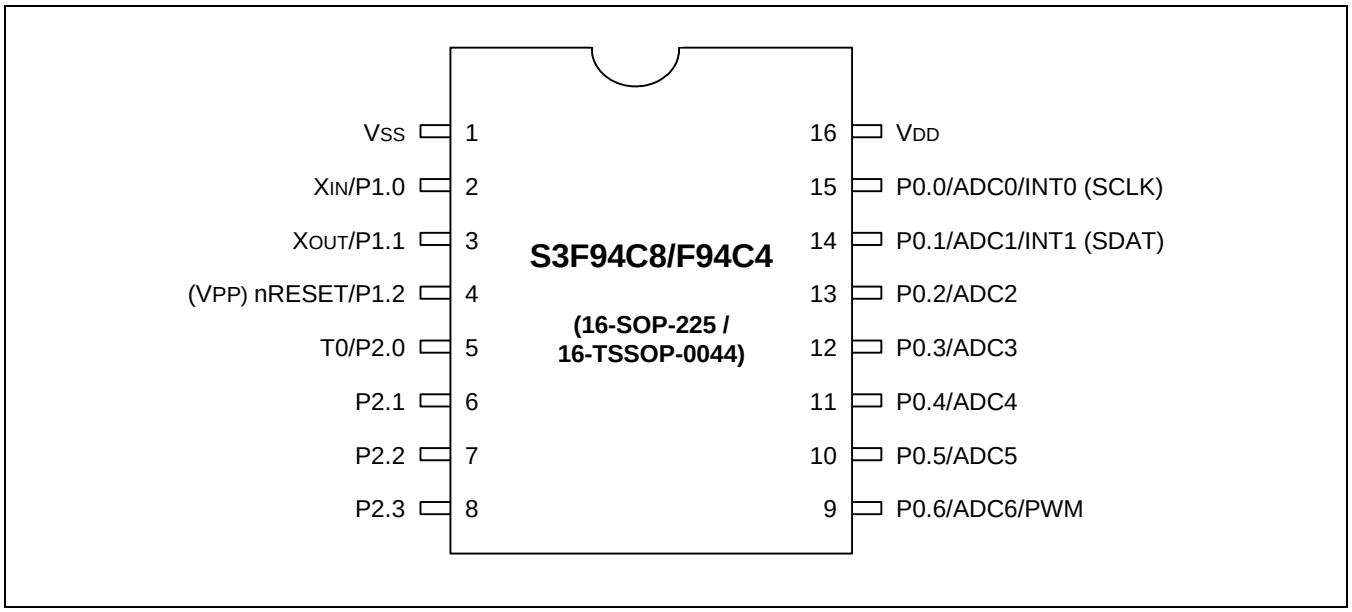


Figure 1-3 Pin Assignment Diagram (16-Pin SOP/TSSOP Package)

1.5 Pin Descriptions

Table 1-1 S3F94C8/F94C4 Pin Descriptions

Pin Name	Input/ Output	Pin Description	Pin Type	Share Pins
P0.0–P0.7	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull output. Pull-up resistors are assigned by software. Port0 pins can also be used as A/D converter input, PWM output, or external interrupt input.	E-1	ADC0–ADC7 INT0/INT1/ PWM
P1.0–P1.1	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistors or pull-down resistors are assigned by software.	E-2	X_{IN}, X_{OUT}
P1.2	I	Schmitt trigger input port	B (NOTE)	RESET
P2.0–P2.6	I/O	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistors are assignable by software.	E	– ADC8/CLO T0
X_{IN}, X_{OUT}	–	Crystal/Ceramic or RC oscillator signal for system clock.		P1.0–P1.1
nRESET	I	Internal LVR or external RESET	B	P1.2
V_{DD}, V_{SS}	–	Voltage input pin and ground		–
CLO	O	System clock output port	E	P2.6
INT0–INT1	I	External interrupt input port	E-1	P0.0, P0.1
PWM	O	14-bit high-speed PWM output	E-1	P0.6
T0	O	Timer0 match output	E-1	P2.0
ADC0–ADC8	I	A/D converter input	E-1 E	P0.0–P0.7 P2.6

NOTE: P1.2 is also used as VPP pin for flash programming. It has an intrinsic internal pull-down resistor, whose value is about 300Kohm.

Table 1-2 Descriptions of Pins Used to Read/Write the Flash ROM

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P0.1	SDAT	18 (20-pin) 14 (16-pin)	I/O	Serial data pin (output when reading, input when writing). Input and push-pull output port can be assigned.
P0.0	SCLK	19 (20-pin), 15 (16-pin)	I	Serial clock pin (input only pin)
RESET/P1.2	VPP	4	I	Power supply pin for entering in the Tool mode. It specifies that MTP has entered in the Tool mode. When 11 V is applied, MTP enters in the Tool mode.
V_{DD}/V_{SS}	V_{DD}/V_{SS}	20 (20-pin), 16 (16-pin) 1 (20-pin), 1 (16-pin)	I	Logic power supply pin.

1.6 Pin Circuits

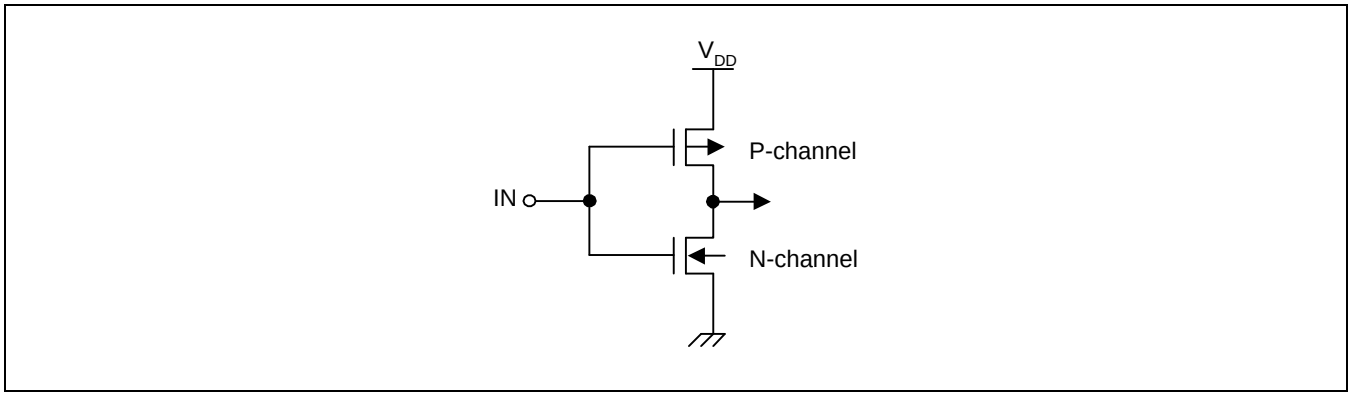


Figure 1-4 Pin Circuit Type A

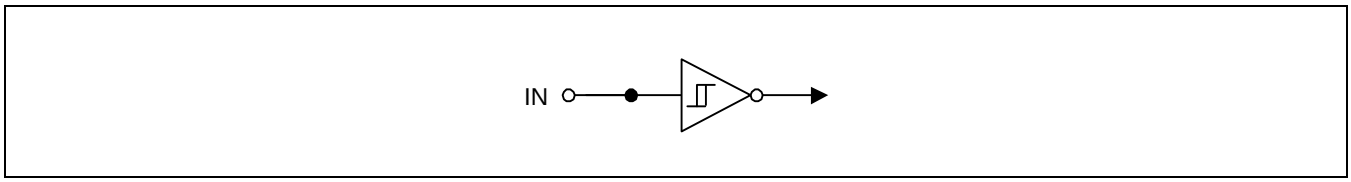


Figure 1-5 Pin Circuit Type B (P1.2)

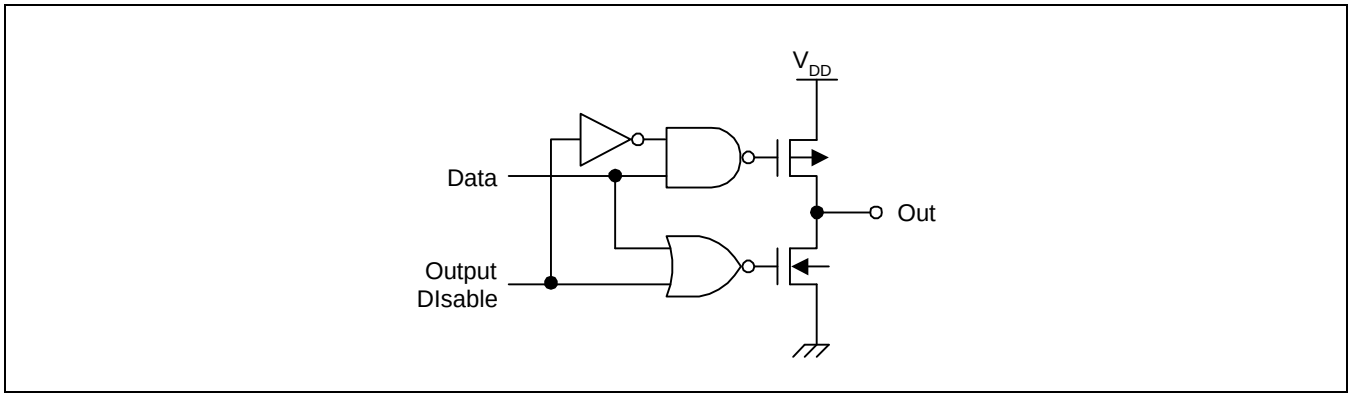


Figure 1-6 Pin Circuit Type C

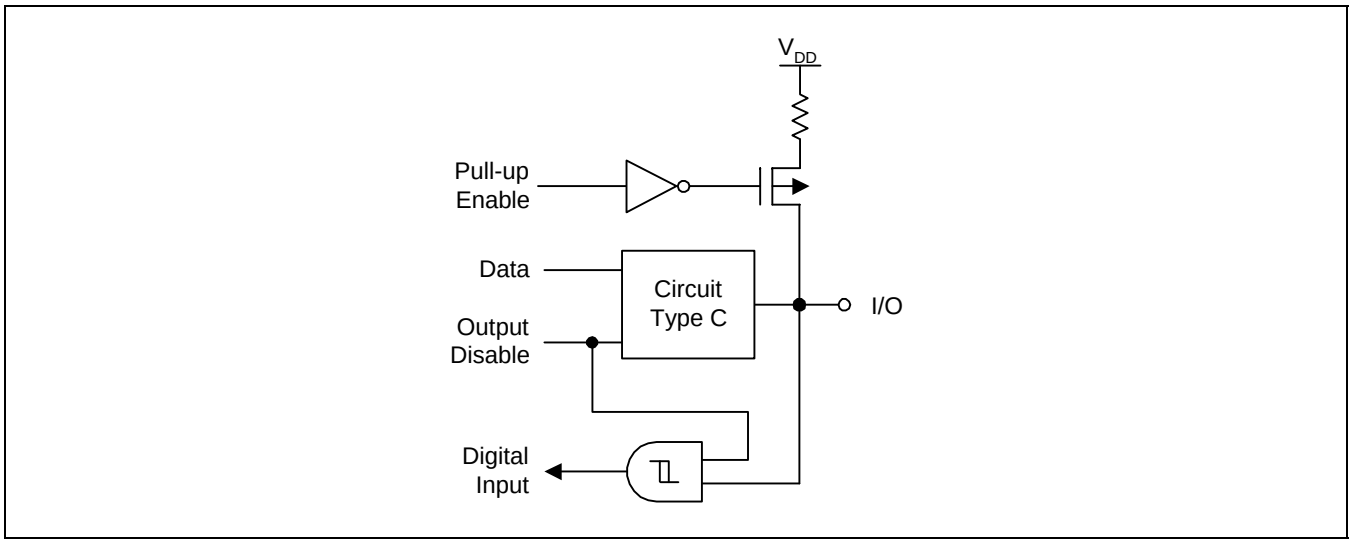


Figure 1-7 Pin Circuit Type D

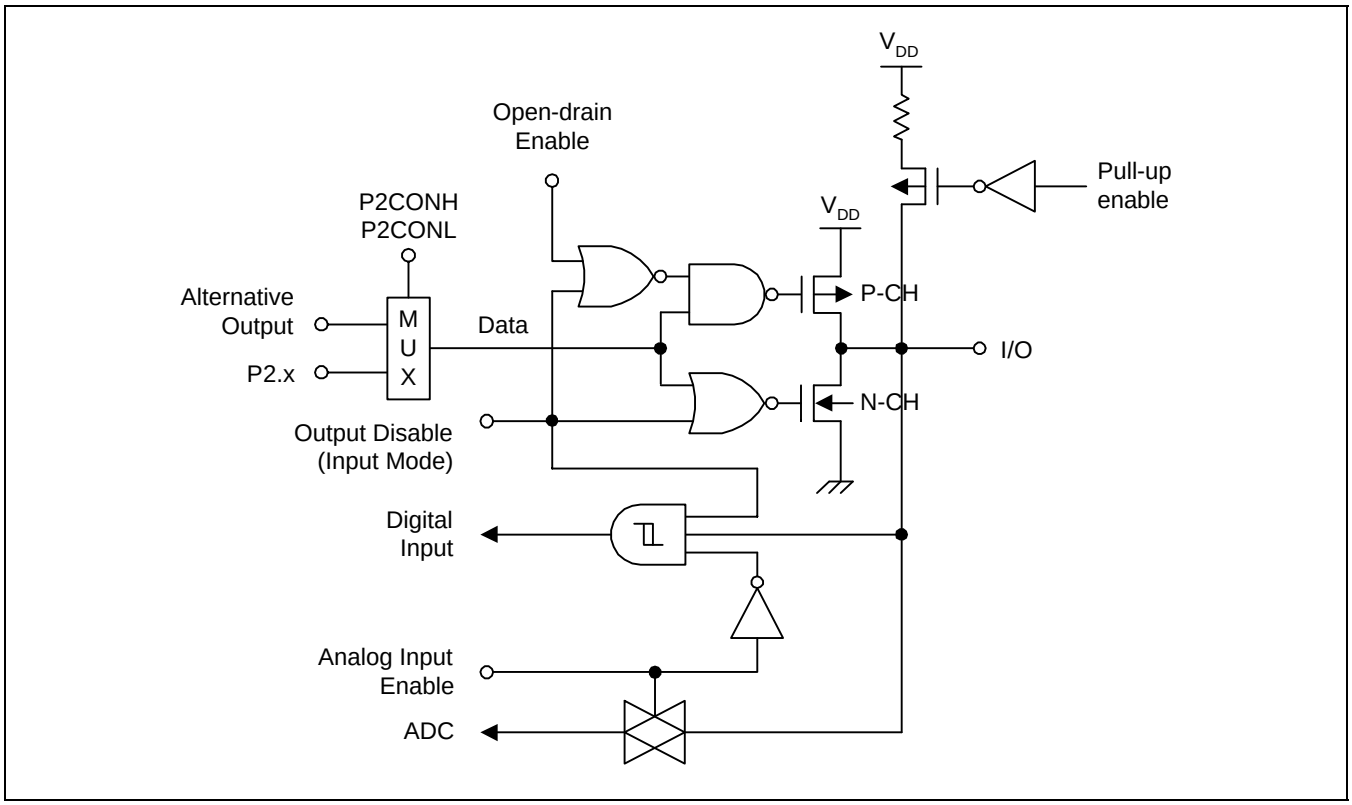


Figure 1-8 Pin Circuit Type E (Port2)

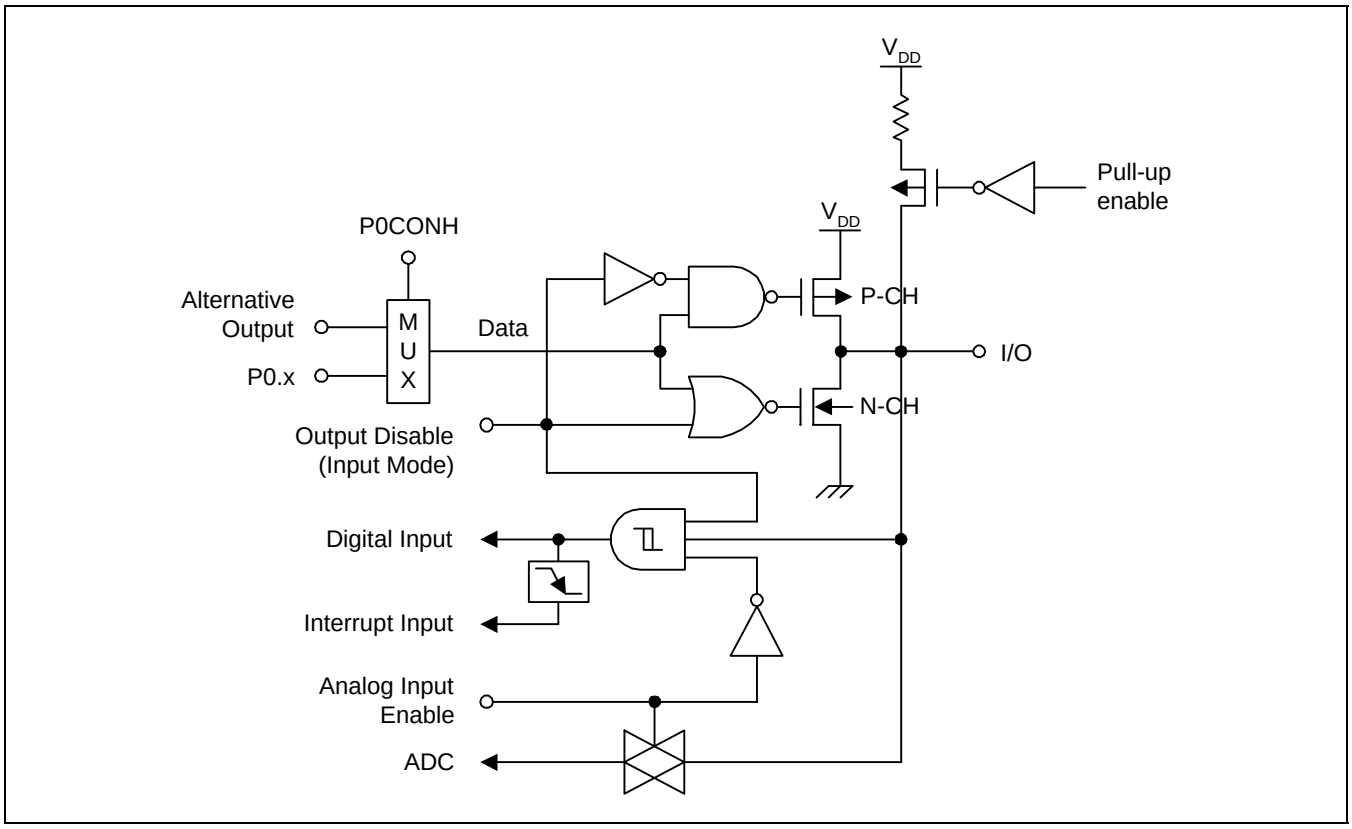


Figure 1-9 Pin Circuit Type E-1 (Port0)

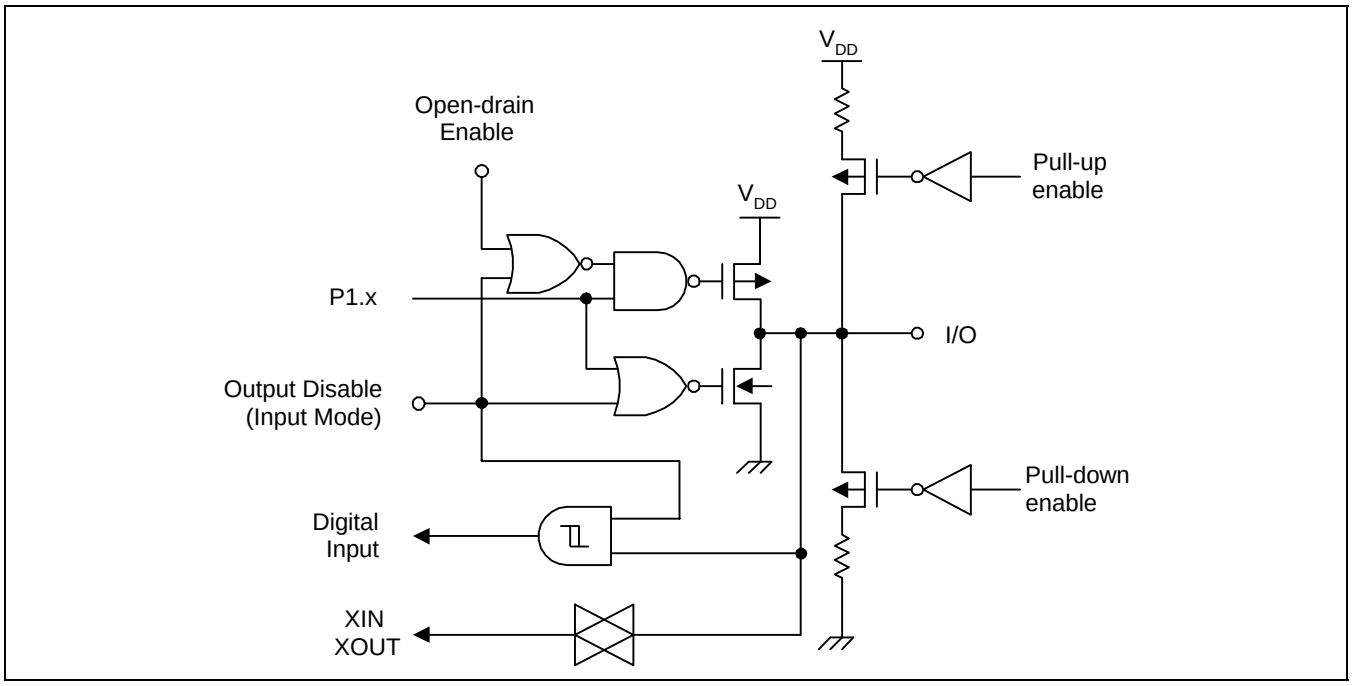


Figure 1-10 Pin Circuit Type E-2 (P1.0-P1.1)

2 ADDRESS SPACES

2.1 Overview of Address Spaces

The S3F94C8/F94C4 microcontroller has two kinds of address space, namely:

Internal full flash program memory (ROM)
Internal register file
A 16-bit address bus supports program memory operations. On the other hand, a separate 8-bit register bus carries addresses and data between the CPU and internal register file.

The S3F94C8/F94C4 microcontroller has 8Kbytes and 4Kbytes of multi-time programmable full flash program memory that is configured in the Internal ROM mode. The chip uses all of the 4K/8K internal program memory.

Also, the S3F94C8/F94C4 microcontroller has 208 general-purpose registers in its internal register file. 32 bytes in the register file are mapped for system and peripheral control functions.

2.2 Program Memory (ROM)

2.2.1 Normal Operating Mode

S3F94C8/F94C4 has 8Kbytes and 4Kbytes of internal multi-time programmable full flash program memory. The program memory address range lies between 0H–1FFFH and 0H-0FFFH.

The first 2-bytes of the ROM (0000H–0001H) specify the interrupt vector address.

Unused locations (0002H–00FFH except 3CH, 3DH, 3EH, and 3FH) can be used as normal program memory. 3CH, 3DH, 3EH, and 3FH are used as smart option ROM cell.

The Program Reset address in ROM is 0100H.

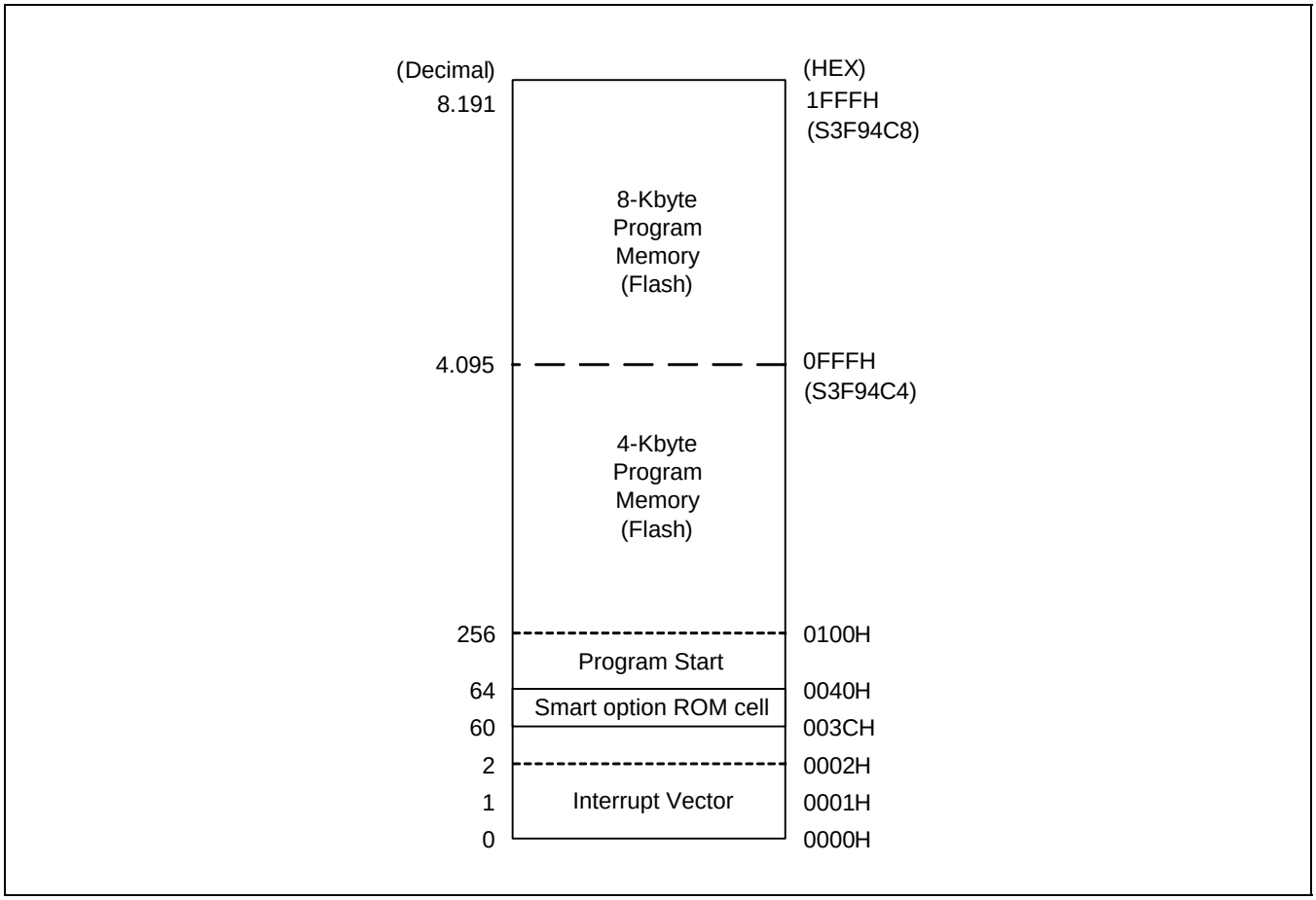
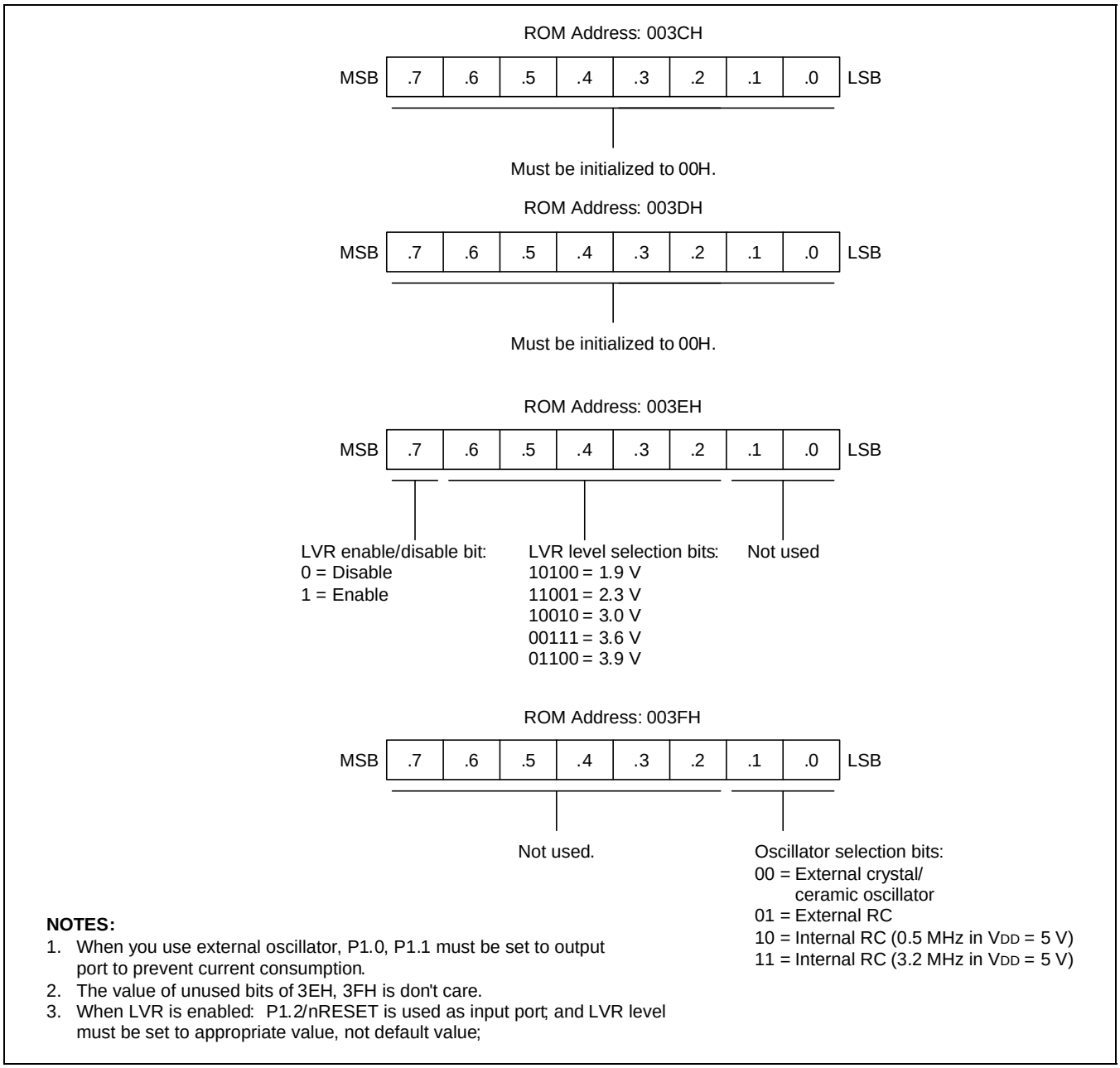


Figure 2-1 Program Memory Address Space

2.2.2 Smart Option

Smart option specifies the ROM option for starting condition of any chip. The ROM addresses used by smart option range from 003CH to 003FH. The S3F94C8/F94C4 only uses 003EH and 003FH. Unused ROM address such as 003CH and 003DH must be initialized to 00H. The default value of ROM 003EH and 003FH is FFH (LVR enable, internal RC oscillator).


ROM Address: 003CH
MSB .7 .6 .5 .4 .3 .2 .1 .0 LSB
Must be initialized to 00H.
ROM Address: 003DH
MSB .7 .6 .5 .4 .3 .2 .1 .0 LSB
Must be initialized to 00H.
ROM Address: 003EH
MSB .7 .6 .5 .4 .3 .2 .1 .0 LSB
LVR enable/disable bit:
0 = Disable
1 = Enable
LVR level selection bits:
10100 = 1.9 V
11001 = 2.3 V
10010 = 3.0 V
00111 = 3.6 V
01100 = 3.9 V
Not used
ROM Address: 003FH
MSB .7 .6 .5 .4 .3 .2 .1 .0 LSB
Not used.
Oscillator selection bits:
00 = External crystal/ ceramic oscillator
01 = External RC
10 = Internal RC (0.5 MHz in VDD = 5 V)
11 = Internal RC (3.2 MHz in VDD = 5 V)

NOTES:
1. When you use external oscillator, P1.0, P1.1 must be set to output port to prevent current consumption.
2. The value of unused bits of 3EH, 3FH is don't care.
3. When LVR is enabled: P1.2/nRESET is used as input port; and LVR level must be set to appropriate value, not default value;

Figure 2-2 Smart Option

Example 2-1 Programming Tip — Smart Option Setting

```
;       << Interrupt Vector Address >>
              ORG    0000H
              Vector 00H, INT_94C8       ; S3F94C8/F94C4 has only one interrupt vector

;       << Smart Option Setting >>
              ORG    003CH
              DB     00H                 ; 003CH must be initialized to 0.
              DB     00H                 ; 003DH must be initialized to 0.
              DB     0E4H                ; 003EH enable LVR (2.3 V)
              DB     03H                 ; 003FH Internal RC (3.2 MHz in VDD = 5 V)

;       << Reset >>
              ORG    0100H
      RESET:  DI
              •
              •
              •
```

2.3 Register Architecture

The upper 64-bytes of internal register file in S3F94C8/F94C4 consist of working registers, system control registers, and peripheral control registers. The lower 192-bytes of internal register file (00H–BFH) are called general-purpose register space. You can access 240 registers in this space; 208 registers are reserved for general-purpose use.

In case of S3F94C8/F94C4, the total number of addressable 8-bit registers is 240. Out of these 240 registers, 32 bytes are used for CPU and system control registers and peripheral control and data registers, 16 bytes are used as shared working registers, and 192 registers are used for general-purpose.

For many SAM88RCRI microcontrollers, the addressable area of internal register file is further expanded by additional register pages in general-purpose register space (00H–BFH: page0). However, this register file expansion is not implemented in S3F94C8/F94C4.

Table 2-1 shows the specific register types and area (in bytes) they occupy in internal register file.

Table 2-1 Register Type Summary

Register Type	Number of Bytes
CPU and system control registers	11
Peripheral, I/O, and clock control and data registers	21
General-purpose registers (including 16-bit common working register area)	208
Total Addressable Bytes	240

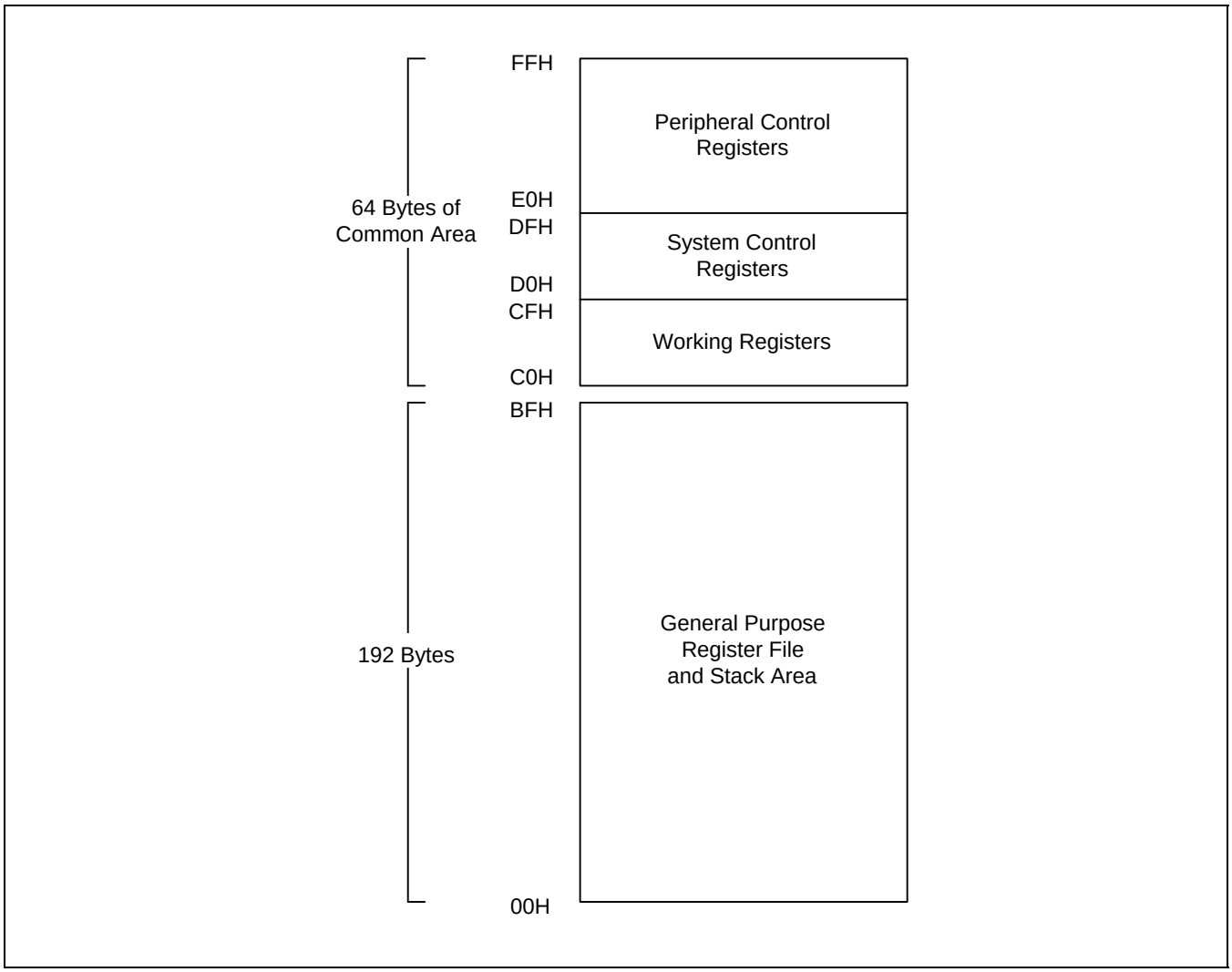


Figure 2-3 Internal Register File Organization

2.4 Common Working Register Area (C0H–CFH)

The SAM88RCRI register architecture provides an efficient method for working register addressing mode. This method takes full advantage of shorter instruction formats to reduce the execution time.

The 16-byte address range is called common area. It can be used as working register to address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations in different pages. However, since the S3F94C8/F94C4 uses only page 0, you can use the common area for any internal data operation.

The working register addressing mode and indirect register addressing mode can be used to access this area.

Registers are addressed as either single 8-bit register or paired 16-bit register. In 16-bit register pairs, the address of the first 8-bit register is always an even number and the address of the next register is an odd number. The most significant byte of the 16-bit data is always stored in the even-numbered register; the least significant byte is always stored in the next (+ 1) odd-numbered register.

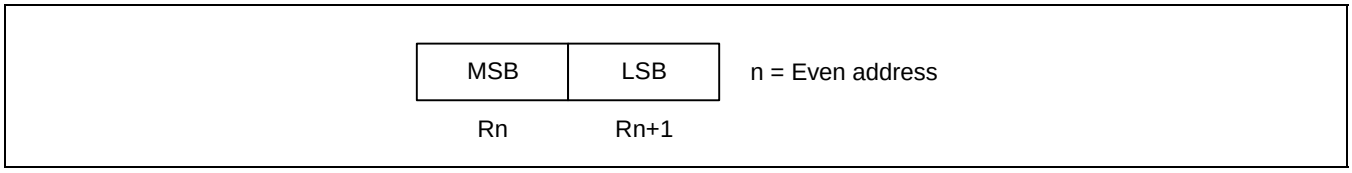


Figure 2-4 16-Bit Register Pairs

Example 2-2 Programming Tip — Addressing the Common Working Register Area

You should access working registers in the common area, C0H–CFH, using working register addressing mode and indirect register addressing mode.

```
1.    LD     0C2H, 40H     ;      Invalid addressing mode!
```
Use working register addressing mode instead.
```
      LD     R2, 40H       ;      R2 (C2H) ← the value in location 40H

2.    ADD    0C3H, #45H    ;      Invalid addressing mode!
```
Use working register addressing mode instead.
```
      ADD    R3, #45H      ;      R3 (C3H) ← R3 + 45H
```

2.5 System Stack

S3F9-series microcontrollers use the system stack for subroutine calls and returns, and for storing data. The PUSH and POP instructions are used to control system stack operations. The S3F94C8/F94C4 architecture supports stack operations in the internal register file.

2.5.1 Stack Operations

Return addresses for procedure calls and interrupts and data are stored on the stack. The contents of the PC are saved on the stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the Programmer Counter (PC) and System Flags (FLAGS) register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address is always decremented before a push operation and incremented after a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of stack, as shown in *Figure 2-5*.

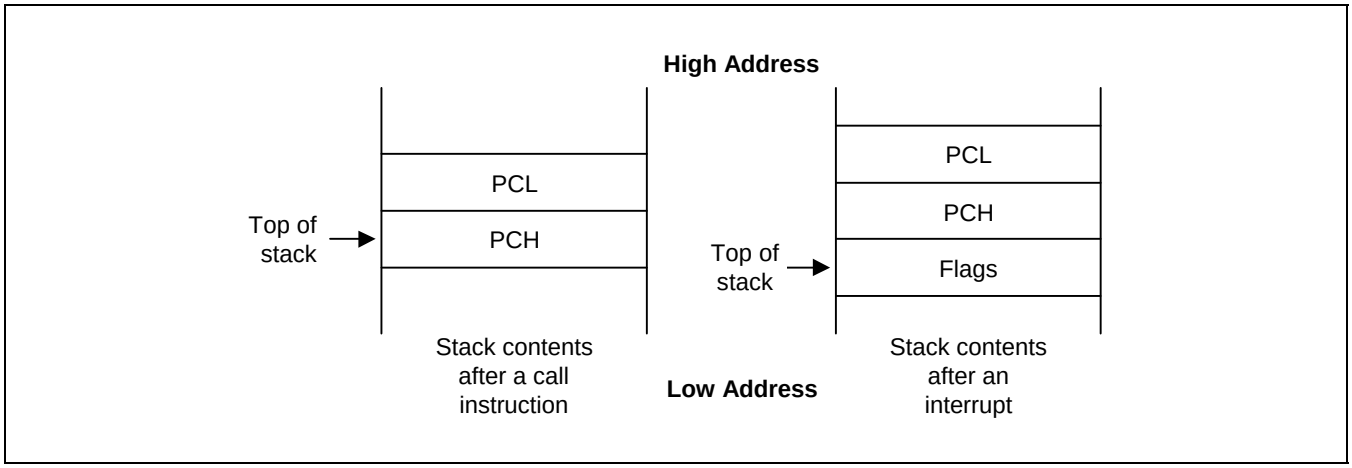


Figure 2-5 Stack Operations

2.5.2 Stack Pointer (SP)

Register location D9H contains 8-bit SP that is used for system stack operations. After reset, the SP value is undetermined.

Since only internal memory 192 bytes space is implemented in S3F94C8/F94C4, the SP must be initialized to an 8-bit value in the range 00H–0C0H.

In case a Stack Pointer is initialized to 00H, it is decreased to FFH when stack operation starts. This means that a Stack Pointer accesses invalid stack area. It is recommended that a stack pointer is initialized to C0H to set upper address of stack to BFH.

Example 2-3 Programming Tip — Standard Stack Operations Using PUSH and POP

The following example shows you how to perform stack operations in the internal register file using PUSH and POP instructions:

```
LD      SP,#0C0H            ;       SP ← C0H (Normally, the SP is set to C0H by the
                            ;       initialization routine)
•
•
•
PUSH    SYM                 ;       Stack address 0BFH   ← SYM
PUSH    R15                 ;       Stack address 0BEH   ← R15
PUSH    20H                 ;       Stack address 0BDH   ← 20H
PUSH    R3                  ;       Stack address 0BCH   ← R3
•
•
•
POP     R3                  ;       R3    ← Stack address 0BCH
POP     20H                 ;       20H   ← Stack address 0BDH
POP     R15                 ;       R15   ← Stack address 0BEH
POP     SYM                 ;       SYM   ← Stack address 0BFH
```

3 ADDRESSING MODES

3.1 Overview of Addressing Modes

Instructions indicate the operation to be performed and the data to be operated upon. Instructions stored in program memory are fetched for execution using the program counter. To determine the location of data operand, Addressing mode is used. The operands specified in SAM88RCRI instructions can include condition codes and immediate data or a location in the register file, program memory, or data memory.

The SAM88RCRI instruction set supports six explicit addressing modes. Not all of these addressing modes are available for each instruction. The addressing modes and their symbols are as follows:

- Register (R)
- Indirect Register (IR)
- Indexed (X)
- Direct Address (DA)
- Relative Address (RA)
- Immediate (IM)

3.1.1 Register Addressing Mode

In Register Addressing mode, the operand specifies the content of a specified register (see *Figure 3-1*). Working register addressing differs from Register Addressing because it uses a 16-byte working register space in the register file and a 4-bit register within that space (see *Figure 3-2*).

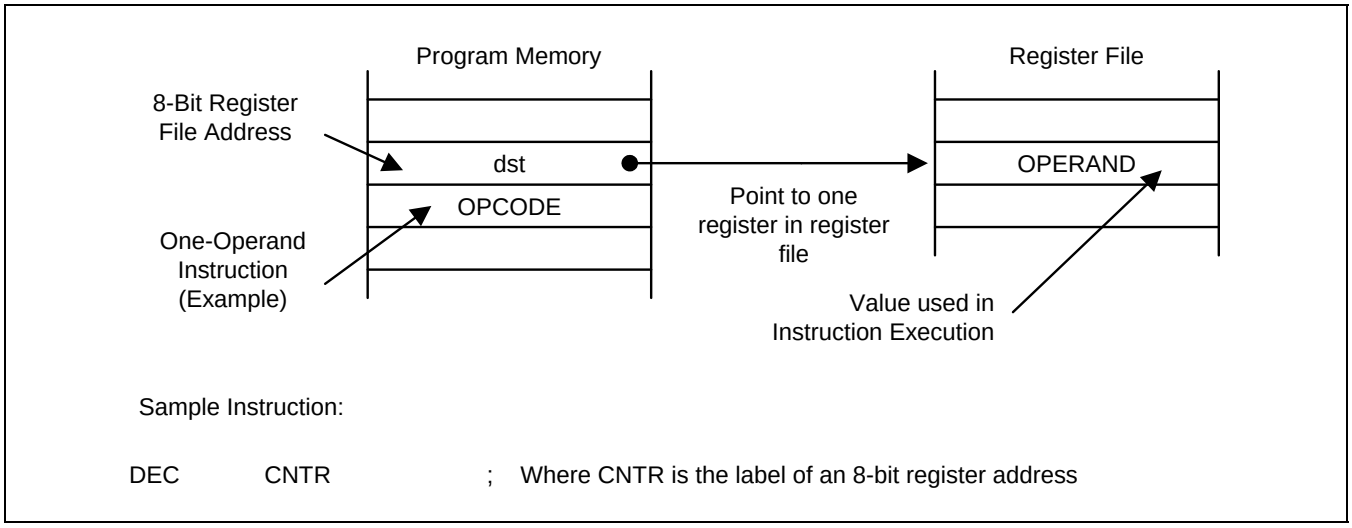


Figure 3-1 Register Addressing

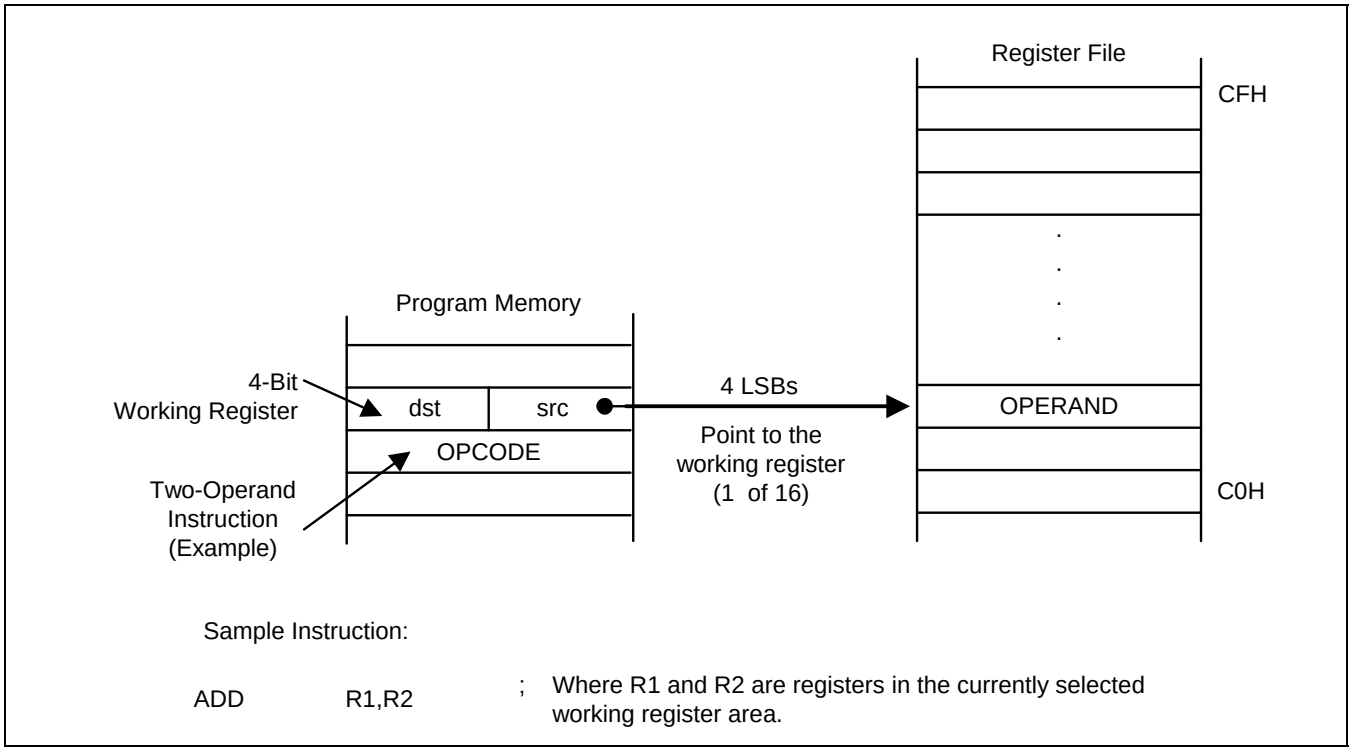
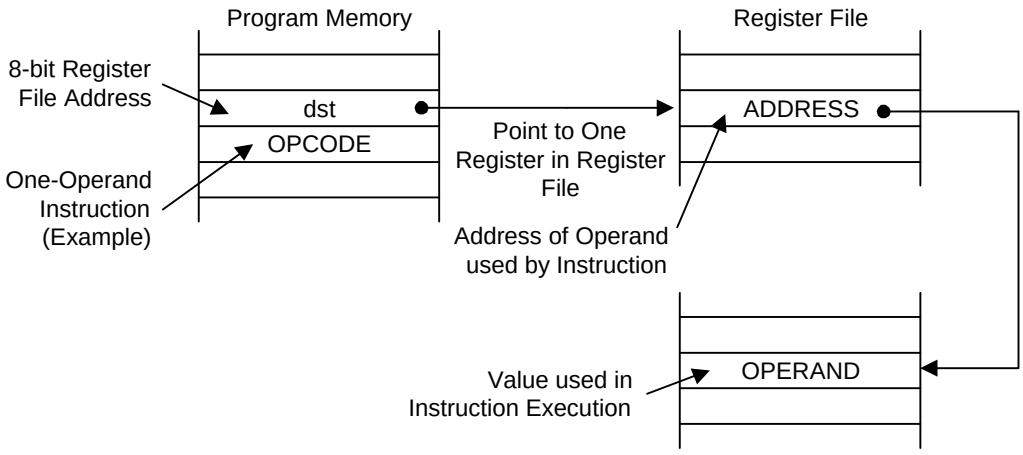


Figure 3-2 Working Register Addressing

3.1.2 Indirect Register Addressing Mode

In Indirect Register (IR) Addressing mode, the contents of specified register or register pair include the address of the operand. Depending on the instruction used, the actual address may point to a register in the register file, to program memory (ROM), or to an external memory space (see *Figure 3-3* ~ *Figure 3-6*).

You can use any 8-bit register to indirectly address another register. Any 16-bit register pair can be used to indirectly address another memory location.



Sample Instruction:

```
RL      @SHIFT          ;   Where SHIFT is the label of an 8-bit register address
```

Figure 3-3 Indirect Register Addressing to Register File

3.1.3 Indirect Register Addressing Mode (1)

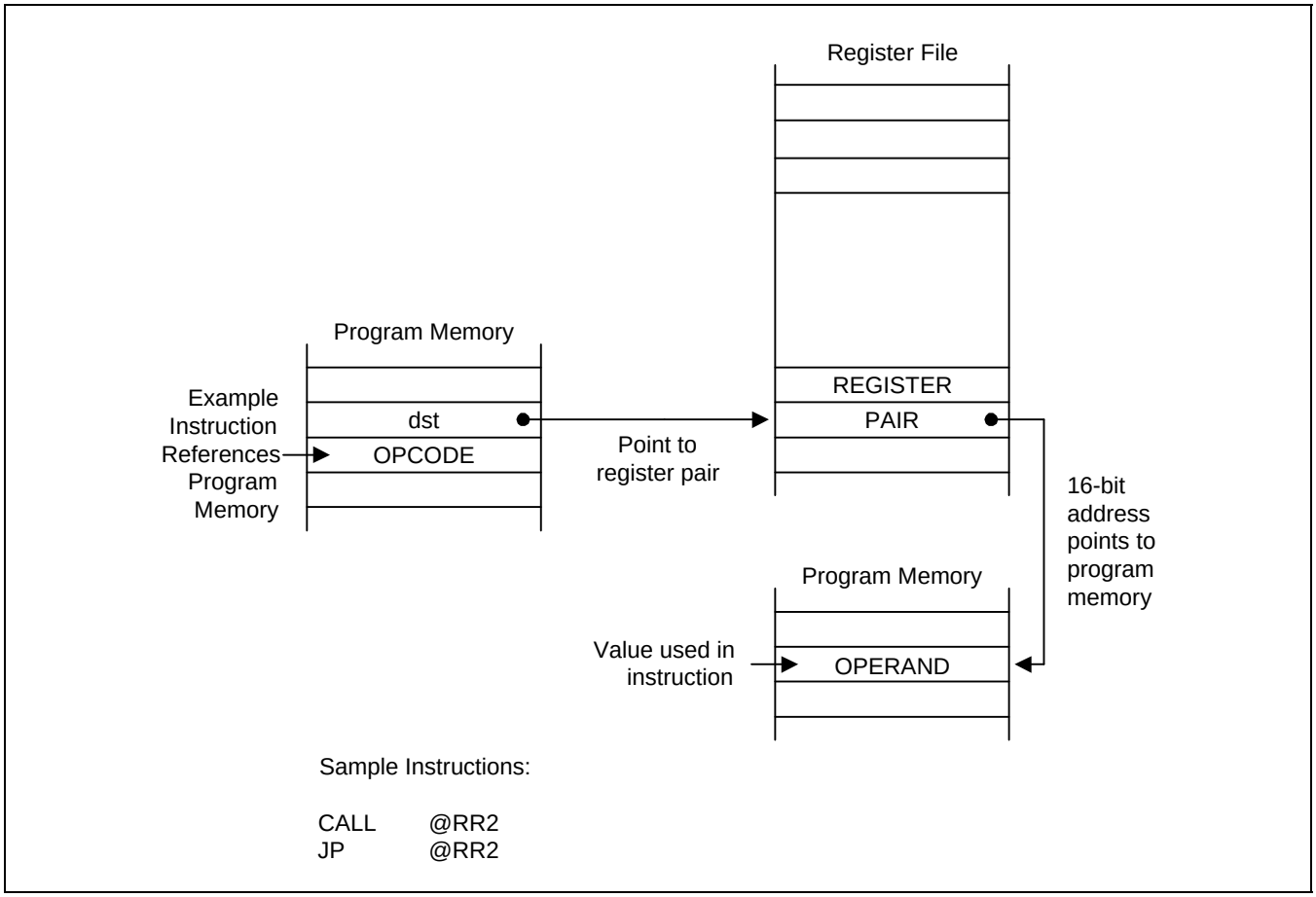


Figure 3-4 Indirect Register Addressing to Program Memory

3.1.4 Indirect Register Addressing Mode (2)

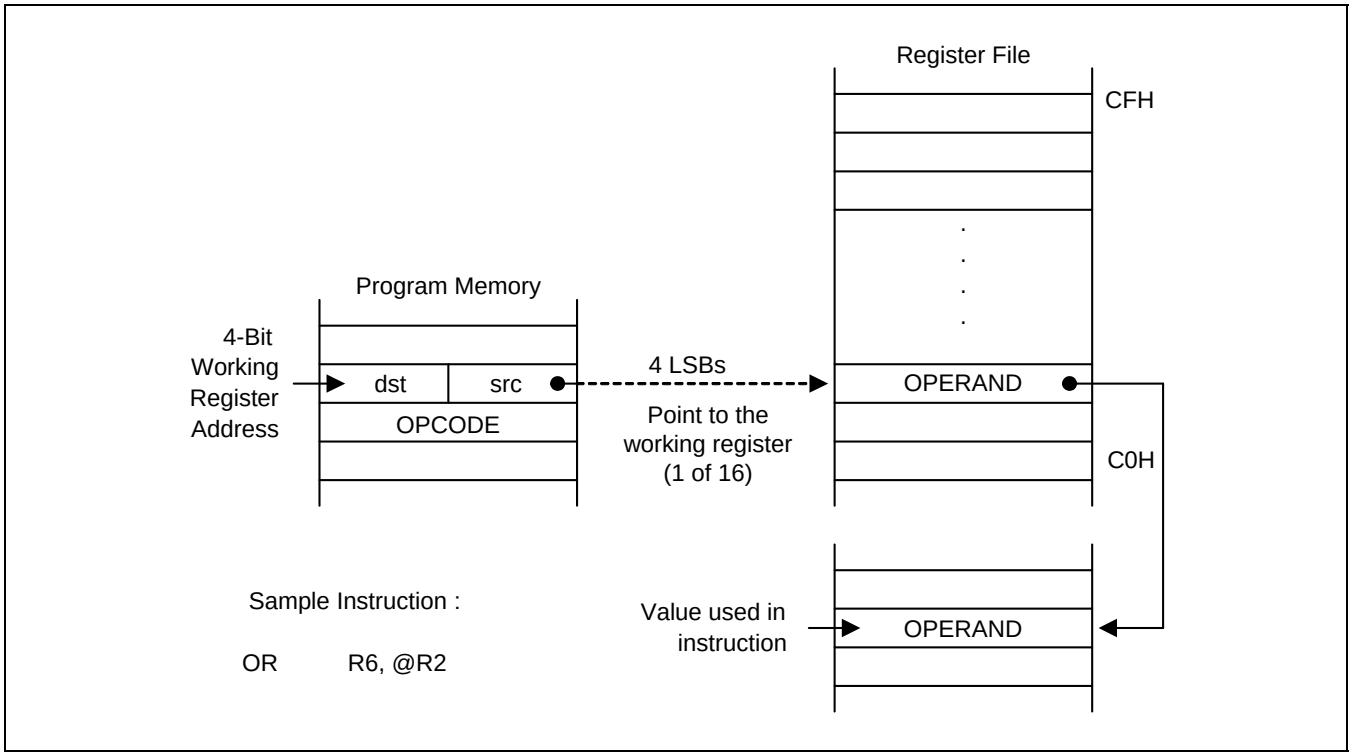


Figure 3-5 Indirect Working Register Addressing to Register File

3.1.5 Indirect Register Addressing Mode (3)

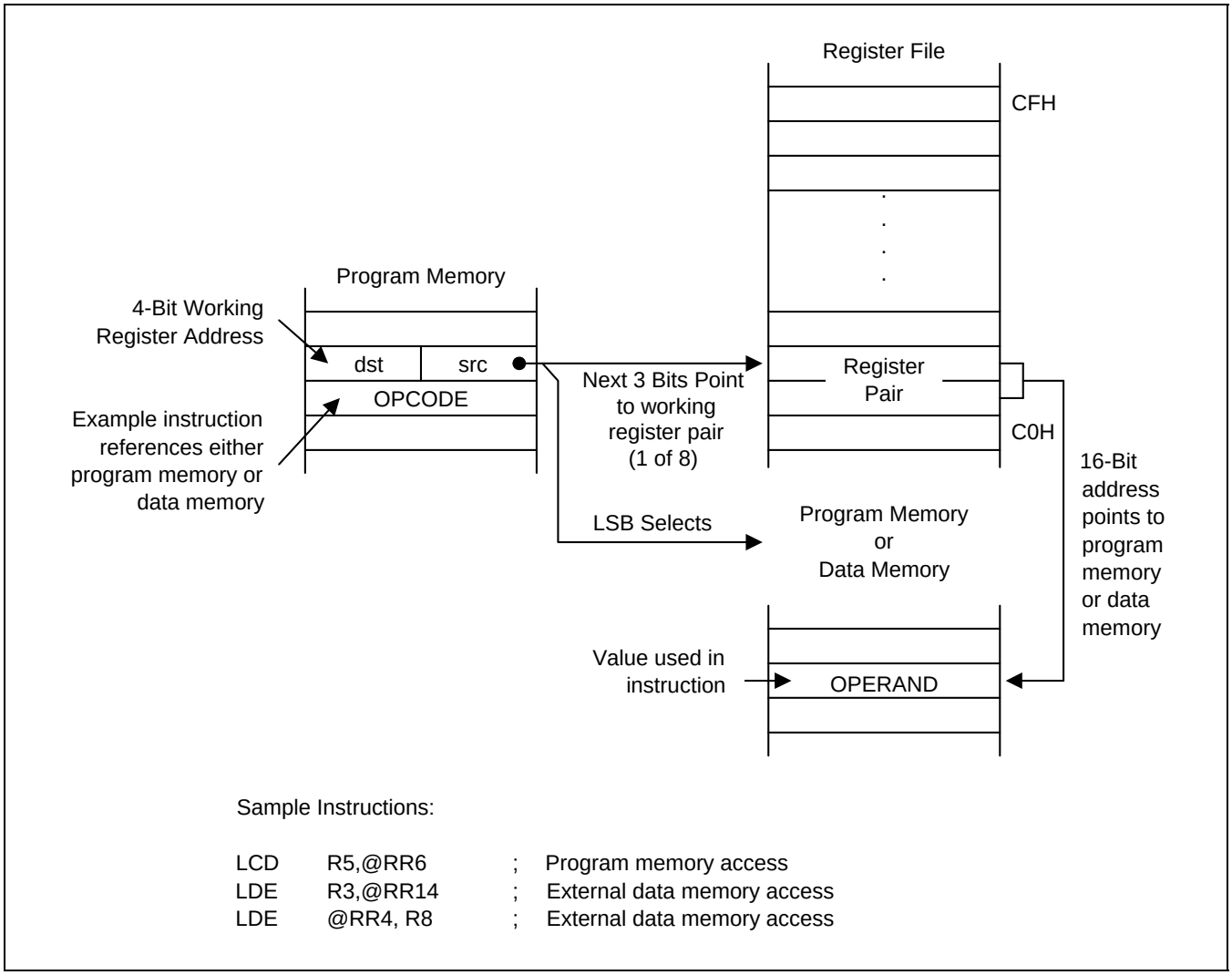


Figure 3-6 Indirect Working Register Addressing to Program or Data Memory

3.1.6 Indexed Addressing Mode

To calculate the effective operand address, Indexed (X) Addressing mode adds an offset value to a base address during instruction execution (see *Figure 3-7*). You can use Indexed Addressing mode to access locations in the internal register file or external memory.

In Short Offset Indexed Addressing mode, the 8-bit displacement is treated as a signed integer in the range of –128 to + 127. This applies to external memory accesses only (see *Figure 3-8*).

For register file addressing, an 8-bit base address provided by the instruction is added to an 8-bit offset contained in a working register. For external memory accesses, the base address is stored in the working register pair designated in the instruction. The 8-bit or 16-bit offset given in the instruction is then added to the base address (see *Figure 3-9*).

The only instruction that supports Indexed Addressing mode for the internal register file is Load instruction (LD). The LDC and LDE instructions support Indexed Addressing mode for internal program memory, external program memory, and for external data memory, when implemented.

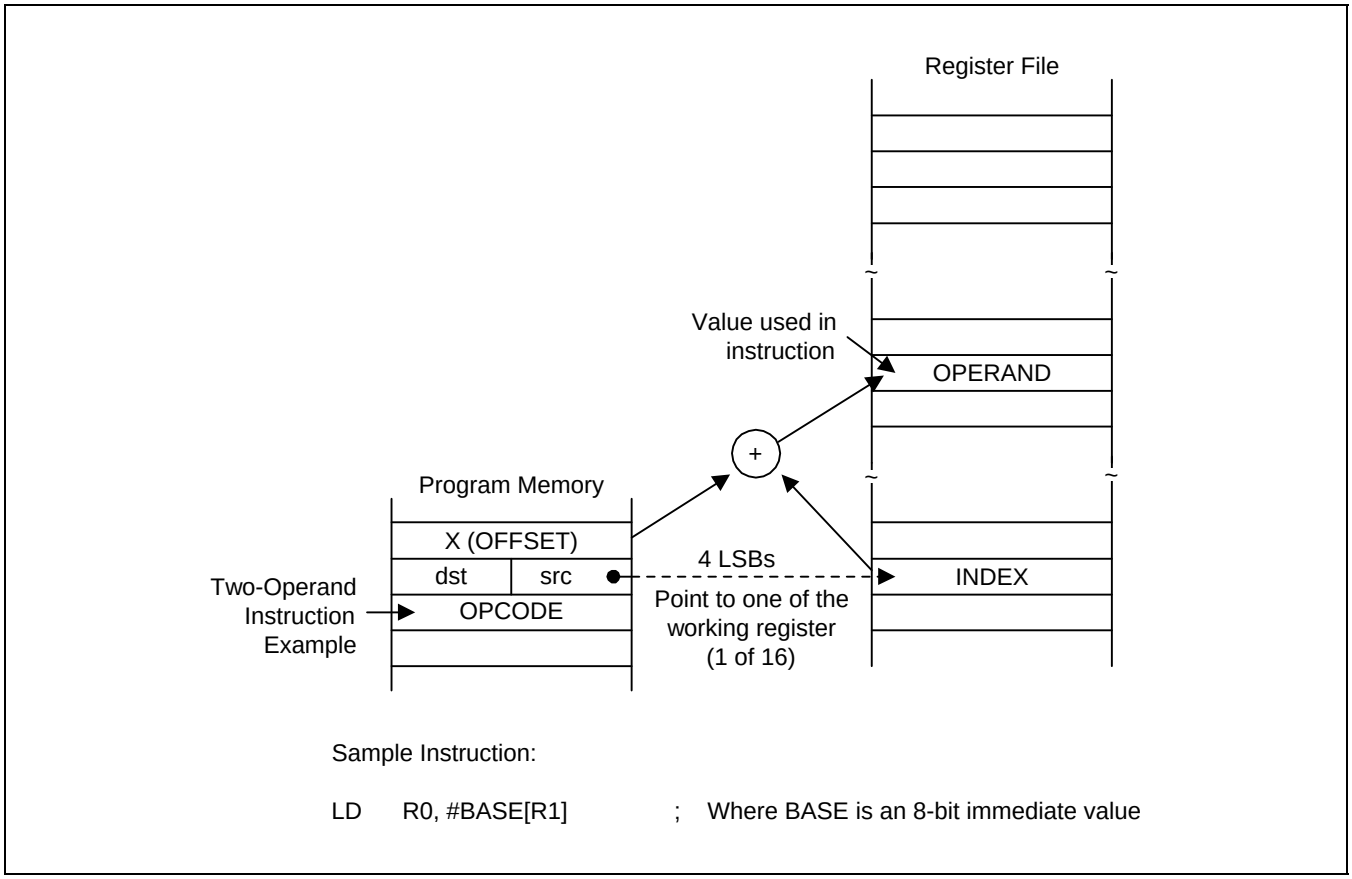
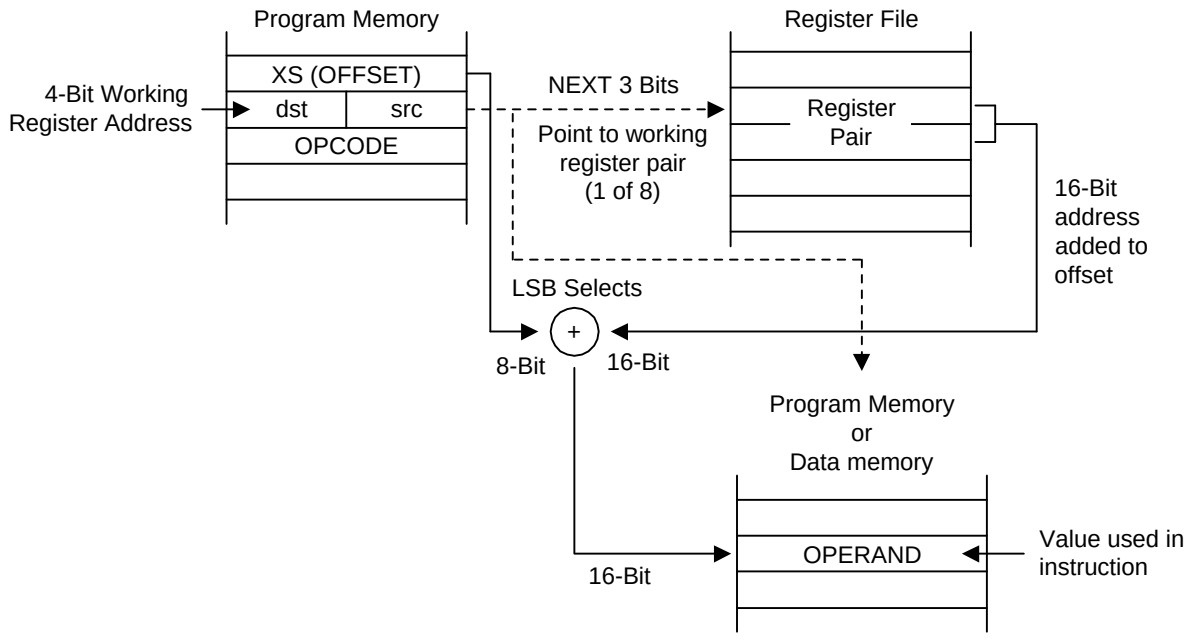


Figure 3-7 Indexed Addressing to Register File

3.1.7 Indexed Addressing Mode (1)

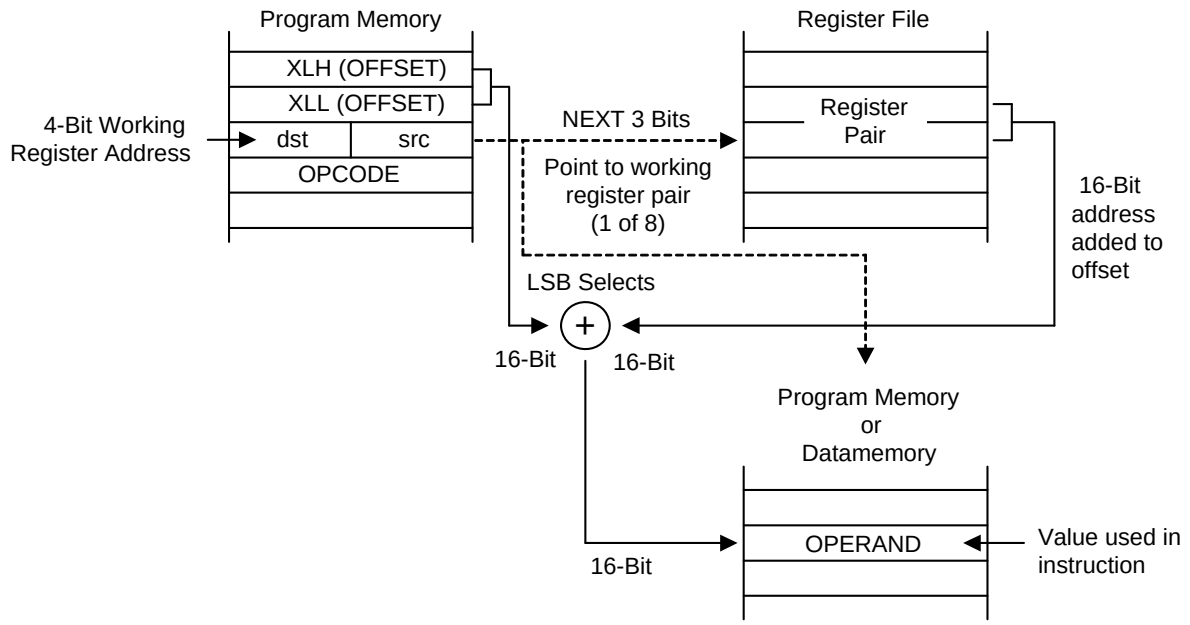


Sample Instructions:

```
LDC     R4, #04H[RR2]     ; The values in the program address (RR2 + #04H)
                          ; are loaded into register R4.
LDE     R4,#04H[RR2]      ; Identical operation to LDC example, except that
                          ; external program memory is accessed.
```

Figure 3-8 Indexed Addressing to Program or Data Memory with Short Offset

3.1.8 Indexed Addressing Mode (2)



Sample Instructions:

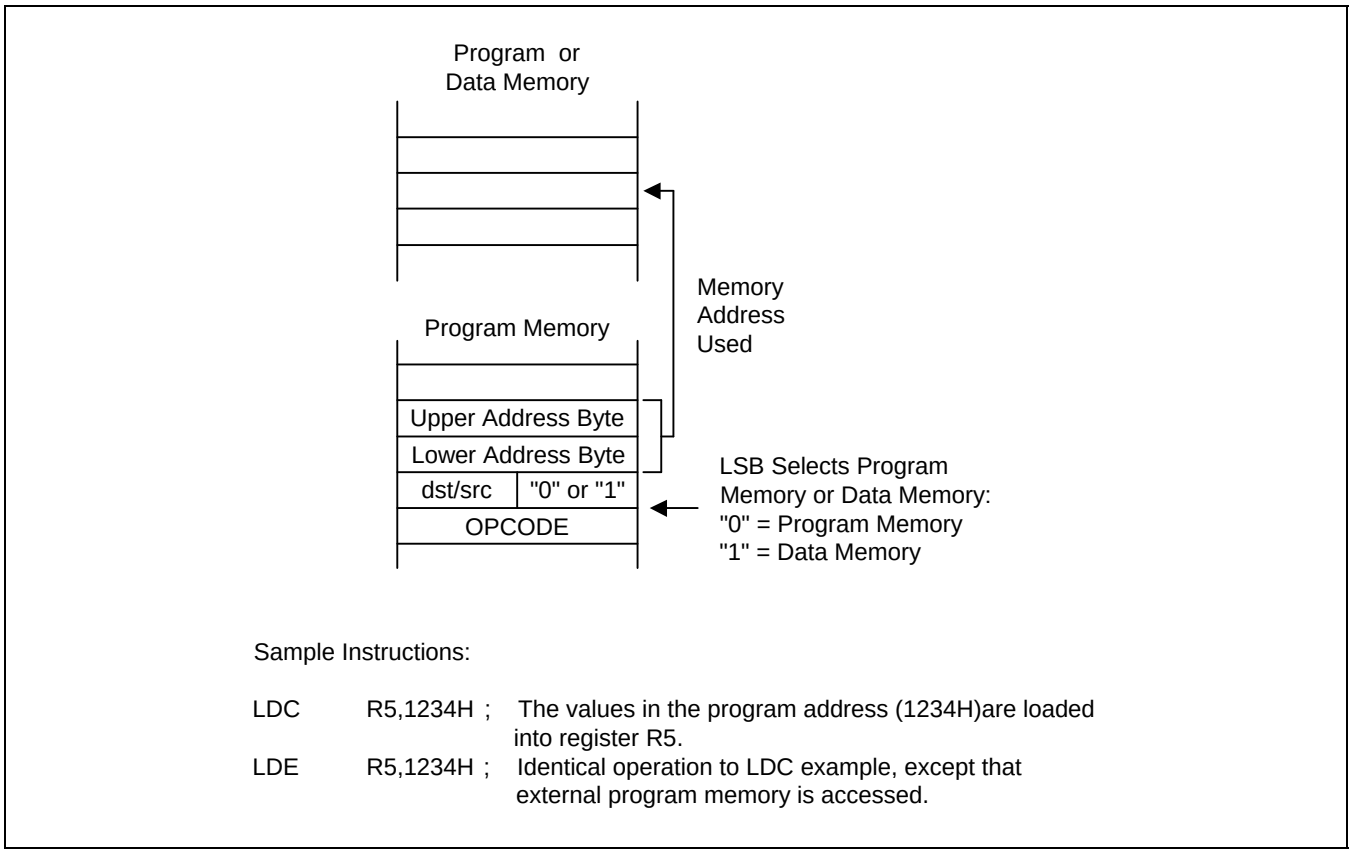
```
LDC     R4, #1000H[RR2]     ;  The values in the program address (RR2 + #1000H)
                               are loaded into register R4.
LDE     R4, #1000H[RR2]     ;  Identical operation to LDC example, except that
                               external program memory is accessed.
```

Figure 3-9 Indexed Addressing to Program or Data Memory with Long Offset

3.1.9 Direct Address Mode

In Direct Address (DA) mode, the instruction provides the operand's 16-bit memory address. Jump (JP) and Call (CALL) instructions use this addressing mode to specify the 16-bit destination address loaded in the Program Counter (PC) whenever a JP or CALL instruction is executed.

The LDC and LDE instructions can use Direct Address mode to specify the source or destination address for Load operations to program memory (LDC) or external data memory (LDE), if implemented.

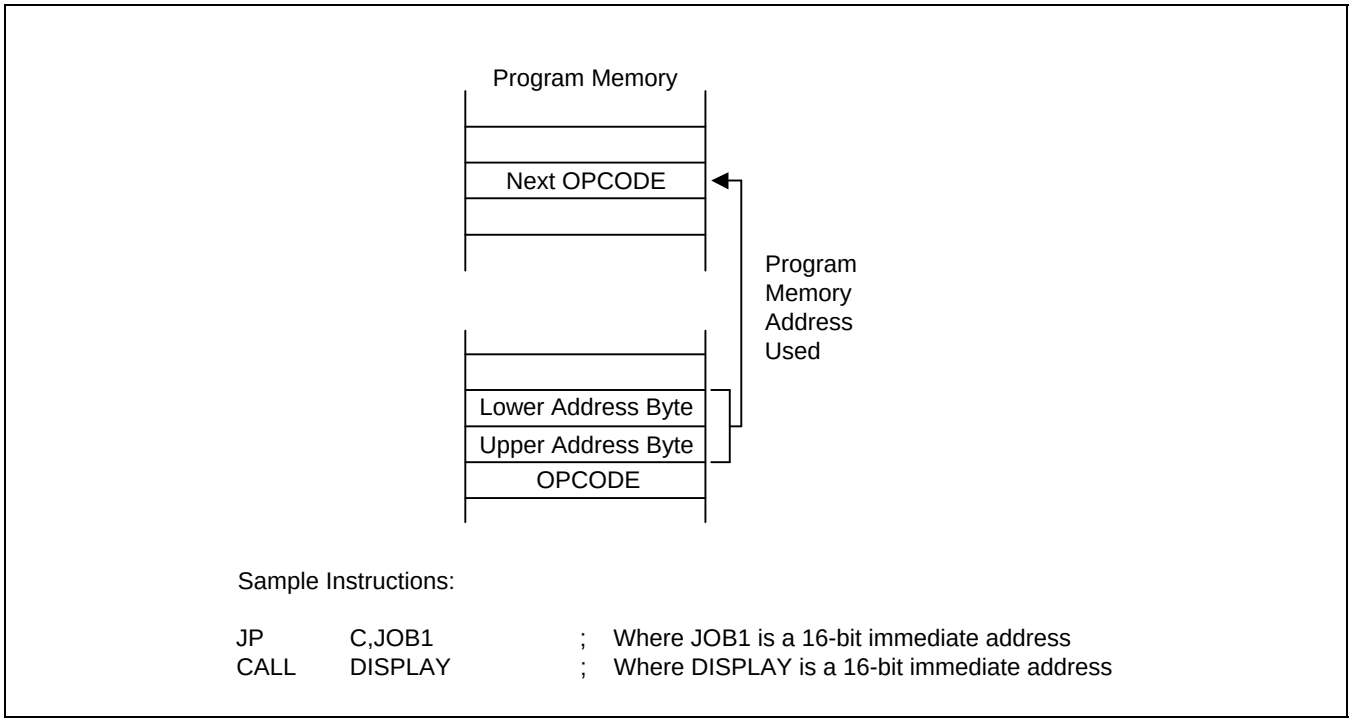


Sample Instructions:

```
LDC     R5,1234H ;  The values in the program address (1234H)are loaded
                    into register R5.
LDE     R5,1234H ;  Identical operation to LDC example, except that
                    external program memory is accessed.
```

Figure 3-10 Direct Addressing for Load Instructions

3.1.10 Direct Address Mode (1)



Sample Instructions:

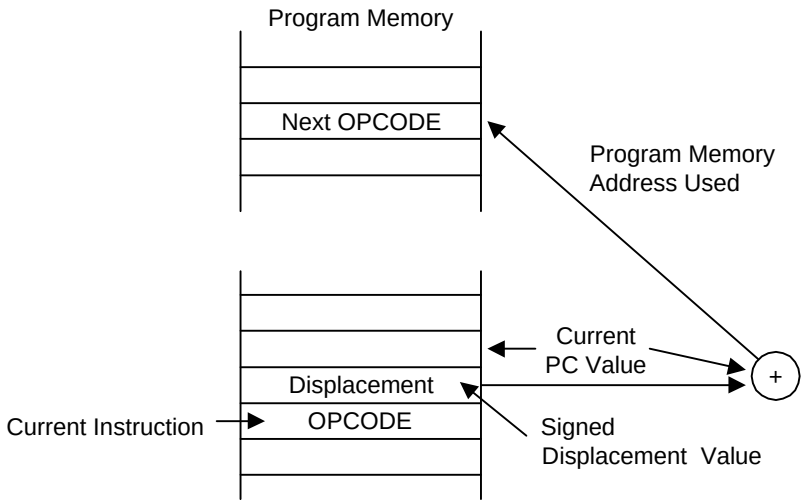
```
JP      C,JOB1      ;  Where JOB1 is a 16-bit immediate address
CALL    DISPLAY     ;  Where DISPLAY is a 16-bit immediate address
```

Figure 3-11 Direct Addressing for Call and Jump Instructions

3.1.11 Relative Address Mode

In Relative Address (RA) mode, a two's-complement signed displacement between – 128 and + 127 is specified in the instruction. The displacement value is then added to the current PC value. The result specifies the address of next instruction to be executed. Before this addition occurs, the PC contains the address of instruction immediately following the current instruction.

The instruction that supports RA addressing is JR.



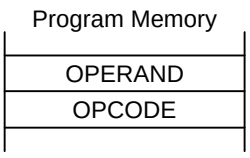
Sample Instructions:

```
JR          ULT,$ + OFFSET          ;   Where OFFSET is a value in the range + 127 to - 128
```

Figure 3-12 Relative Addressing

3.1.12 Immediate Mode

In Immediate (IM) Addressing mode, the operand value used in instruction is the value supplied in operand field itself. Immediate addressing mode is useful for loading constant values into registers.



(The Operand value is in the instruction)

Sample Instruction: LD R0,#0AAH

Figure 3-13 Immediate Addressing

4 ADDRESSING REGISTERS

4.1 Overview of Control Registers

In this section, detailed description of the S3F94C8/F94C4 control registers has been provided. This will help you familiarize with the mapped locations in register file. You can also use this description for quick reference while writing application programs.

Table 4-1 summarizes the system and peripheral registers. In addition, *Figure 4-1* illustrates the important features of standard register description format.

Control register descriptions are arranged in alphabetical order according to register mnemonic. For more information about control registers, refer to the context of various peripheral hardware description in Part II of this manual.

Table 4-1 System and Peripheral Control Registers

Register Name	Mnemonic	Address and Location		RESET value (Bit)							
		Address	R/W	7	6	5	4	3	2	1	0
Timer 0 Counter Register	T0CNT	D0H	R	0	0	0	0	0	0	0	0
Timer 0 Data Register	T0DATA	D1H	R/W	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	D2H	R/W	0	0	–	–	0	–	0	0
Location D3H Is Not Mapped											
Clock Control Register	CLKCON	D4H	R/W	0	–	–	0	0	–	–	–
System Flags Register	FLAGS	D5H	R/W	x	x	x	x	–	–	–	–
Locations D6H–D8H Are Not Mapped											
Stack Pointer Register	SP	D9H	R/W	x	x	x	x	x	x	x	x
Location DAH Is Not Mapped											
MDS Special Register	MDSREG	DBH	R/W	0	0	0	0	0	0	0	0
Basic Timer Control Register	BTCON	DCH	R/W	0	0	0	0	0	0	0	0
Basic Timer Counter	BTCNT	DDH	R	0	0	0	0	0	0	0	0
Test Mode Control Register	FTSTCON	DEH	W	–	–	0	0	0	0	0	0
System Mode Register	SYM	DFH	R/W	–	–	–	–	0	0	0	0
Port 0 Data Register	P0	E0H	R/W	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	E1H	R/W	–	–	–	–	–	0	0	0
Port 2 Data Register	P2	E2H	R/W	–	0	0	0	0	0	0	0
Locations E3H–E5H Are Not Mapped											
Port 0 Control Register (High Byte)	P0CONH	E6H	R/W	0	0	0	0	0	0	0	0
Port 0 Control Register	P0CONL	E7H	R/W	0	0	0	0	0	0	0	0
Port 0 Interrupt Pending Register	P0PND	E8H	R/W	–	–	–	–	0	0	0	0
Port 1 Control Register	P1CON	E9H	R/W	0	0	–	–	0	0	0	0
Port 2 Control Register (High Byte)	P2CONH	EAH	R/W	–	0	0	0	0	0	0	0
Port 2 Control Register (Low Byte)	P2CONL	EBH	R/W	0	0	0	0	0	0	0	0
Flash Memory Control Register	FMCON	ECH	R/W	0	0	0	0	–	–	–	0
Flash Memory User Programming Enable Register	FMUSR	EDH	R/W	0	0	0	0	0	0	0	0
Flash Memory Sector Address Register (High Byte)	FMSECH	EEH	R/W	0	0	0	0	0	0	0	0
Flash Memory Sector Address Register (Low Byte)	FMSECL	EFH	R/W	0	0	0	0	0	0	0	0
PWM Data Register 1	PWMDATA1	F0H	R/W	0	0	0	0	0	0	0	0
PWM Extension Register	PWMEX	F1H	R/W	0	0	0	0	0	0	0	0
PWM Data Register	PWMDATA	F2H	R/W	0	0	0	0	0	0	0	0

Register Name	Mnemonic	Address and Location		RESET value (Bit)							
		Address	R/W	7	6	5	4	3	2	1	0
PWM Control Register	PWMCON	F3H	R/W	0	0	–	0	0	0	0	0
STOP Control Register	STOPCON	F4H	R/W	0	0	0	0	0	0	0	0
Locations F5H–F6H Are Not Mapped											
A/D Control Register	ADCON	F7H	R/W	0	0	0	0	0	0	0	0
A/D Converter Data Register (High)	ADDATAH	F8H	R	x	x	x	x	x	x	x	x
A/D Converter Data Register (Low)	ADDATAL	F9H	R	0	0	0	0	0	0	x	x
Locations FAH–FFH Are Not Mapped											

NOTE:

1. –: Not mapped or not used, x: Undefined
2. The register, FTSTCON, is not used. Its value should always be ‘00H’ during normal operation.

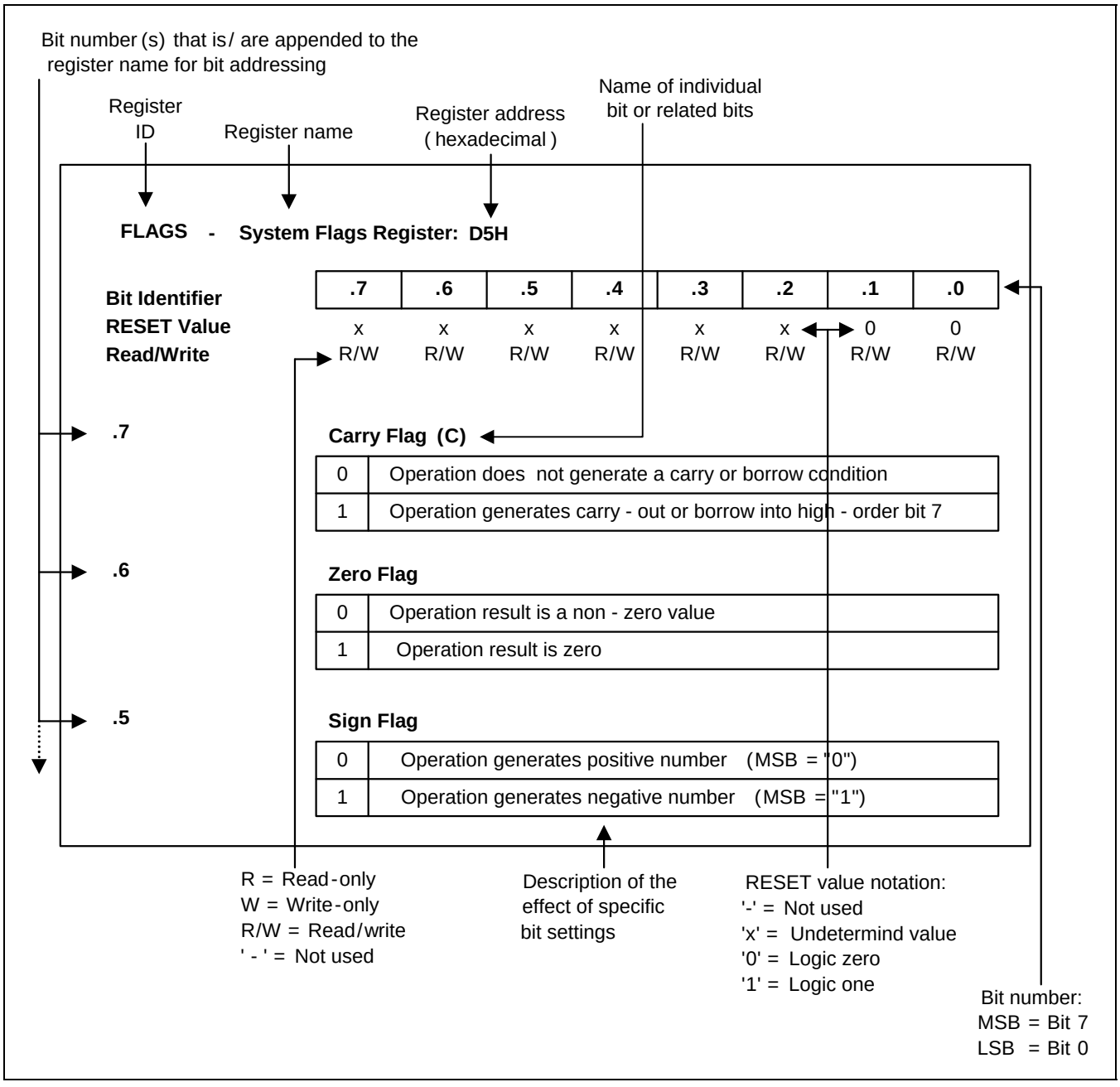

Bit number (s) that is/ are appended to the register name for bit addressing
Register ID
Register name
Register address (hexadecimal)
Name of individual bit or related bits
FLAGS - System Flags Register: D5H
Bit Identifier
RESET Value
Read/Write
.7 .6 .5 .4 .3 .2 .1 .0
x x x x x x 0 0
R/W R/W R/W R/W R/W R/W R/W R/W
.7
Carry Flag (C)
0 Operation does not generate a carry or borrow condition
1 Operation generates carry - out or borrow into high - order bit 7
.6
Zero Flag
0 Operation result is a non - zero value
1 Operation result is zero
.5
Sign Flag
0 Operation generates positive number (MSB = "0")
1 Operation generates negative number (MSB = "1")
R = Read-only
W = Write-only
R/W = Read/write
' - ' = Not used
Description of the effect of specific bit settings
RESET value notation:
'-' = Not used
'x' = Undetermind value
'0' = Logic zero
'1' = Logic one
Bit number:
MSB = Bit 7
LSB = Bit 0

Figure 4-1 Register Description Format

4.1.1 ADCON — A/D Converter Control Register: F7H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.4 A/D Converter Input Pin Selection Bits

0	0	0	0	ADC0 (P0.0)
0	0	0	1	ADC1 (P0.1)
0	0	1	0	ADC2 (P0.2)
0	0	1	1	ADC3 (P0.3)
0	1	0	0	ADC4 (P0.4)
0	1	0	1	ADC5 (P0.5)
0	1	1	0	ADC6 (P0.6)
0	1	1	1	ADC7 (P0.7)
1	0	0	0	ADC8 (P2.6)
1	0	0	1	Connected with GND internally
1	0	1	0	Connected with GND internally
1	0	1	1	Connected with GND internally
1	1	0	0	Connected with GND internally
1	1	0	1	Connected with GND internally
1	1	1	0	Connected with GND internally
1	1	1	1	Connected with GND internally

.3 End-of-Conversion Status Bit

0	A/D conversion is in progress
1	A/D conversion is complete

.2–.1 Clock Source Selection Bit (NOTE)

0	0	fOSC/16 (fOSC ≤ 10 MHz)
0	1	fOSC/8 (fOSC ≤ 10 MHz)
1	0	fOSC/4 (fOSC ≤ 10 MHz)
1	1	fOSC/1 (fOSC ≤ 4 MHz)

.0 Conversion Start Bit

0	No effect
1	A/D conversion start

NOTE: Maximum ADC Clock Input = 4 MHz.

4.1.2 BTCON — Basic Timer Control Register: DCH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.4 Watchdog Timer Function Enable Bit

1	0	1	0	Disables watchdog timer function
Others				Enables watchdog timer function

.3–.2 Basic Timer Input Clock Selection Code

0	0	fOSC/4096
0	1	fOSC/1024
1	0	fOSC/128
1	1	Invalid setting

.1 Basic Timer 8-Bit Counter Clear Bit

0	No effect
1	Clears the basic timer counter value

.0 Basic Timer and Timer 0 Divider Clear Bit

0	No effect
1	Clears both dividers

NOTE: When you write a “1” to BTCON.0 (or BTCON.1), the basic timer divider and timer 0 divider (basic timer counter) are cleared. The bit is then cleared automatically to “0”.

4.1.3 CLKCON — Clock Control Register: D4H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	–	–	0	0	–	–	–
Read/Write	R/W	–	–	R/W	R/W	–	–	–

.7 Oscillator IRQ Wake-up Function Enable Bit

0	Enables IRQ for main system oscillator wake-up function
1	Disables IRQ for main system oscillator wake-up function

.6–.5 Not used for S3F94C8/F94C4

.4–.3 Divided by Selection Bits for CPU Clock frequency

0	0	Divide by 16 (fOSC/16)
0	1	Divide by 8 (fOSC/8)
1	0	Divide by 2 (fOSC/2)
1	1	Non-divided clock (fOSC)

.2–.0 Not used for S3F94C8/F94C4

4.1.4 FLAGS — System Flags Register: D5H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	–	–	–	–
Read/Write	R/W	R/W	R/W	R/W	–	–	–	–

.7 Carry Flag (C)

0	Operation does not generate a carry or borrow condition
1	Operation generates a carry-out or borrow into high-order bit 7

.6 Zero Flag (Z)

0	Operation result is a non-zero value
1	Operation result is zero

.5 Sign Flag (S)

0	Operation generates a positive number (MSB = “0”)
1	Operation generates a negative number (MSB = “1”)

.4 Overflow Flag (V)

0	Operation result is $\leq$ + 127 or $\geq$ – 128
1	Operation result is > + 127 or < – 128

.3–.0 Not used for S3F94C8/F94C4

NOTE: The unused bits .3-.0 should always be kept as ‘0’ in normal operation; otherwise, it may cause error.

4.1.5 FMCON — Flash Memory Control Register: ECH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	–	–	–	0
Read/Write	R/W	R/W	R/W	R/W	–	–	–	R/W

.7–.4 Flash Memory Mode Selection Bits

0	1	0	1	Programming mode
1	0	1	0	Sector erase mode
0	1	1	0	Hard lock mode
Other values				Not available

.3–.1

Not used for the S3F94C8/F94C4

.0 Flash Operation Start Bit

0	Operation stops
1	Operation starts (This bit will be cleared automatically just after the corresponding operation is completed).

4.1.6 FMSECH — Flash Memory Sector Address Register (High Byte): EEH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.0 Flash Memory Sector Address Bits (High Byte)

The 15th - 8th bits select a sector of flash ROM

NOTE: The high-byte flash memory sector address pointer value specifies the higher eight bits of 16-bit pointer address.

4.1.7 FMSECL — Flash Memory Sector Address Register (Low Byte): EFH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7 Flash Memory Sector Address Bit (Low Byte)

The 7th bit selects a sector of flash ROM

.6–.0 Bits 6–0

Don’t care

NOTE: The low-byte flash memory sector address pointer value specifies the lower eight bits of 16-bit pointer address.

4.1.8 FMUSR — Flash Memory User Programming Enable Register: EDH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
Reset Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.0 Flash Memory User Programming Enable Bits

1 0 1 0 0 1 0 1	Enables user programming mode
Other values	Disables user programming mode

4.1.9 P0CONH — Port 0 Control Register (High Byte): E6H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.6 Port 0, P0.7/ADC7 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	A/D converter input (ADC7); Schmitt trigger input off

.5–.4 Port 0, P0.6/ADC6/PWM Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Alternative function (PWM output)
1	0	Push-pull output
1	1	A/D converter input (ADC6); Schmitt trigger input off

.3–.2 Port 0, P0.5/ADC5 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	A/D converter input (ADC5); Schmitt trigger input off

.1–.0 Port 0, P0.4/ADC4 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	A/D converter input (ADC4); Schmitt trigger input off

4.1.10 P0CONL — Port 0 Control Register (Low Byte): E7H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.6 Port 0, P0.3/ADC3 Configuration Bits

0	0	Schmitt trigger input
0	1	Schmitt trigger input; pull-up enable
1	0	Push-pull output
1	1	A/D converter input (ADC3); Schmitt trigger input off

.5–.4 Port 0, P0.2/ADC2 Configuration Bits

0	0	Schmitt trigger input
0	1	Schmitt trigger input; pull-up enable
1	0	Push-pull output
1	1	A/D converter input (ADC2); Schmitt trigger input off

.3–.2 Port 0, P0.1/ADC1/INT1 Configuration Bits

0	0	Schmitt trigger input/falling edge interrupt input
0	1	Schmitt trigger input; pull-up enable/falling edge interrupt input
1	0	Push-pull output
1	1	A/D converter input (ADC1); Schmitt trigger input off

.1–.0 Port 0, P0.0/ADC0/INT0 Configuration Bits

0	0	Schmitt trigger input/falling edge interrupt input
0	1	Schmitt trigger input; pull-up enable/falling edge interrupt input
1	0	Push-pull output
1	1	A/D converter input (ADC0); Schmitt trigger input off

4.1.11 P0PND — Port 0 Interrupt Pending Register: E8H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	0	0	0	0
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W

.7–.4

Not used for the S3F94C8/F94C4

.3 Port 0.1/ADC1/INT1 Interrupt Enable Bit

0	Disables INT1 falling edge interrupt
1	Enables INT1 falling edge interrupt

.2 Port 0.1/ADC1/INT1 Interrupt Pending Bit

0	No interrupt pending (when read)
0	Clears pending bit (when write)
1	Interrupt is pending (when read)
1	No effect (when write)

.1 Port 0.0/ADC0/INT0 Interrupt Enable Bit

0	Disables INT0 falling edge interrupt
1	Enables INT0 falling edge interrupt

.0 Port 0.0/ADC0/INT0 Interrupt Pending Bit

0	No interrupt pending (when read)
0	Clears pending bit (when write)
1	Interrupt pending (when read)
1	No effect (when write)

4.1.12 P1CON — Port 1 Control Register: E9H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	–	–	0	0	0	0
Read/Write	R/W	R/W	–	–	R/W	R/W	R/W	R/W

.7 Part 1.1 N-channel open-drain Enable Bit

0	Configures P1.1 as a push-pull output
1	Configures P1.1 as an n-channel open-drain output

.6 Port 1.0 N-channel open-drain Enable Bit

0	Configures P1.0 as a push-pull output
1	Configures P1.0 as an n-channel open-drain output

.5–.4 Not used for S3F94C8/F94C4

.3–.2 Port 1, P1.1 Interrupt Pending Bits

0	0	Schmitt trigger input
0	1	Schmitt trigger input; pull-up enable
1	0	Output
1	1	Schmitt trigger input; pull-down enable

.1–.0 Port 1, P1.0 Configuration Bits

0	0	Schmitt trigger input;
0	1	Schmitt trigger input; pull-up enable
1	0	Output
1	1	Schmitt trigger input; pull-down enable

NOTE: When you use external oscillator, P1.0 and P1.1 must be set to output port to prevent current consumption.

4.1.13 P2CONH — Port 2 Control Register (High Byte): EAH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	0	0	0	0	0	0	0
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7 Not used for the S3F94C8/F94C4

.6–.4 Port 2, P2.6/ADC8/CLO Configuration Bits

0	0	0	Schmitt trigger input; pull-up enable
0	0	1	Schmitt trigger input
0	1	x	ADC input
1	0	0	Push-pull output
1	0	1	Open-drain output; pull-up enable
1	1	0	Open-drain output
1	1	1	Alternative function; CLO output

.3–.2 Port 2, 2.5 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.1–.0 Port 2, 2.4 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

NOTE: To avoid problems resulting from noise, do not use CLO output.

4.1.14 P2CONL — Port 2 Control Register (Low Byte): EBH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.6 Part 2, P2.3 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.5–.4 Port 2, P2.2 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.3–.2 Port 2, P2.1 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	Open-drain output

.1–.0 Port 2, P2.0 Configuration Bits

0	0	Schmitt trigger input; pull-up enable
0	1	Schmitt trigger input
1	0	Push-pull output
1	1	T0 match output

4.1.15 PWMCON — PWM Control Register: F3H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	–	0	0	0	0	0
Read/Write	R/W	R/W	–	R/W	R/W	R/W	R/W	R/W

.7–.6 PWM Input Clock Selection Bits

0	0	fOSC/64
0	1	fOSC/8
1	0	fOSC/2
1	1	fOSC/1

.5

Not used for S3F94C8/F94C4

.4 PWMDATA Reload Interval Selection Bit

0	Reloads from extension up counter overflow
1	Reloads from base up counter overflow

.3 PWM Counter Clear Bit

0	No effect
1	Clears the PWM counter (when write)

.2 PWM Counter Enable Bit

0	Stops counter
1	Starts (Resumes countering)

.1 PWM Overflow Interrupt Enable Bit (8-Bit Overflow)

0	Disables interrupt
1	Enables interrupt

.0 PWM Overflow Interrupt Pending Bit

0	No interrupt is pending (when read)
0	Clears pending bit (when write)
1	Interrupt is pending (when read)
1	No effect (when write)

NOTE:

1. PWMCON.3 is not auto-cleared. You must pay attention when clearing pending bit. (Refer to pages 11-12 for more information).
2. PWMCON.5 should always be set to ‘0’.

4.1.16 PWMEX — PWM Extension Register: F1H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.2 PWM Extension Bits

PWM extension bits for 6+6 resolution and 8+6 resolution; not used in 6+2 resolution.

.1–.0 PWM Base/extension Control bits:

0	0	Base 6-bit (PWMDATA.7-.2) + Extension 2-bit (PWMDATA.1-.0)
1	0	
0	1	Base 6-bit (PWMDATA1.5-.0) + Extension 6-bit (PWMEX.7-.2)
1	1	Base 8-bit (PWMDATA1.7-.0) + Extension 6-bit (PWMEX.7-.2)

4.1.17 STOPCON — STOP Mode Control Register: E4H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

.7–.0 Watchdog Timer Function Enable Bit

10100101	Enables STOP instruction
Other value	Disables STOP instruction

NOTE: When STOPCON register does not have #0A5H value and you use STOP instruction, program counter (PC) is changed to reset address.

4.1.18 SYM — System Mode Register: DFH

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	0	0	0	0
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W

.7–.4

Not used for S3F94C8/F94C4

.3 Global Interrupt Enable Bit

0	Disables all interrupts
1	Enables all interrupt

.2–.0 Page Select Bits

0	0	0	Page 0
0	0	1	Page 1 (Not used for S3F94C8/F94C4)
0	1	0	Page 2 (Not used for S3F94C8/F94C4)
0	1	1	Page 3 (Not used for S3F94C8/F94C4)

4.1.19 T0CON — TIMER 0 Control Register: D2H

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	–	–	0	–	0	0
Read/Write	R/W	R/W	–	–	R/W	–	R/W	R/W

.7–.6 Timer 0 Input Clock Selection Bits

0	0	fOSC/4096
0	1	fOSC/256
1	0	fOSC/8
1	1	fOSC/1

.5–.4

Not used for S3F94C8/F94C4

.3 Timer 0 Counter Clear Bit

0	No effect
1	Clears the timer 0 counter (when write)

.2

Not used for S3F94C8/F94C4

.1 Timer 0 Interrupt Enable Bit

0	Disables interrupt
1	Enables interrupt

.0 Timer 0 Interrupt Pending Bit (Match interrupt)

0	No interrupt is pending (when read)
0	Clears pending bit (when write)
1	Interrupt is pending (when read)
1	No effect (when write)

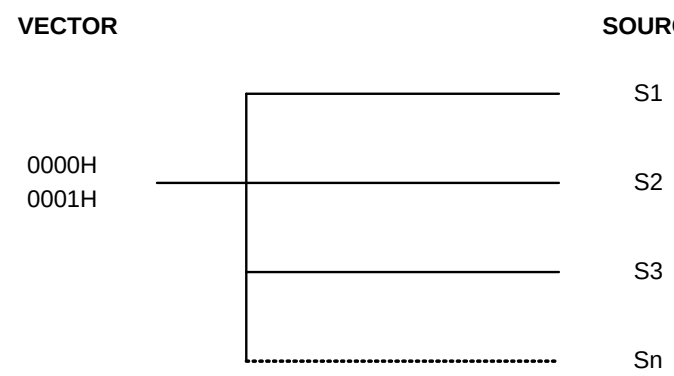
NOTE:

1. T0CON.3 is not auto-cleared. You must pay attention while clearing pending bit. (Refer to pages 10-12 for more information.)
2. To use T0 match output, set T0CON.3 to “1”. (Refer to pages 10-7 for more information.)

5 INTERRUPT STRUCTURE

5.1 Overview of Interrupt Structure

The SAM88RCRI interrupt structure has two basic components, namely, vector and sources. The number of interrupt sources can be serviced through an interrupt vector that is assigned in ROM address 0000H.



NOTES:
1. The SAM88 RCRI interrupt has only one vector address (0000H-0001H).
2. The number of Sn value is expandable.

Figure 5-1 S3F9-Series Interrupt Type

5.1.1 Interrupt Processing Control Points

Interrupt processing can be controlled in two ways, that is, either globally or by specific interrupt source. The system-level control points in the interrupt structure are as follows:

Global interrupt enable and disable (by EI and DI instructions)
Interrupt source enable and disable settings in the corresponding peripheral control register(s)

5.1.2 Enable/Disable Interrupt Instructions (EI, DI)

The system mode register, SYM (DFH), is used to enable and disable interrupt processing. SYM.3 is the enable and disable bit for global interrupt processing.

The system initialization routine executed after a reset must always contain an EI instruction to globally enable the interrupt structure.

Although you can manipulate SYM.3 directly to enable and disable interrupts during normal operation, it is recommended that you use the EI and DI instructions for this purpose.

5.1.3 Interrupt Pending Function Types

When the interrupt service routine (ISR) has executed, the application program's service routine must clear the appropriate pending bit before return from interrupt subroutine (IRET) occurs.

5.1.4 Interrupt Priority

Since there is no interrupt priority register in SAM88RCRI, the order of service is determined by a sequence of source executed in interrupt service routine.

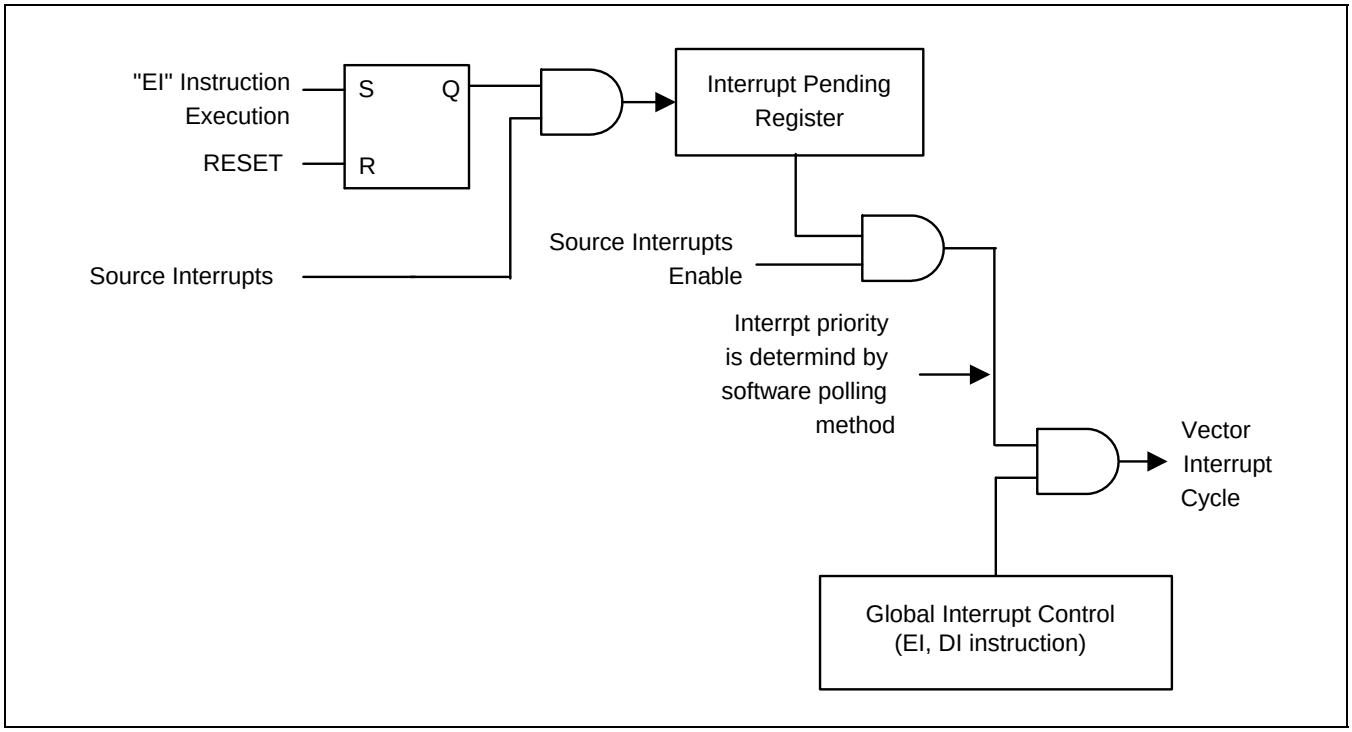


Figure 5-2 Interrupt Function Diagram

5.1.5 Interrupt Source Service Sequence

The interrupt request polling and servicing sequence is as follows:

1. Set the interrupt request pending bit to “1”. The source generates an interrupt.
2. The CPU generates an interrupt acknowledge signal.
3. The service routine starts.
4. Clear the pending flag of source to “0” using the software.
5. Determine the interrupt priority by using software polling method.

5.1.6 Interrupt Service Routines

Before an interrupt request can be serviced, the following conditions must be met:

Interrupt processing must be enabled (EI, SYM.3 = “1”)
Interrupt must be enabled at the interrupt’s source (peripheral control register)
If all of the above conditions are met, the interrupt request is acknowledged at the end of instruction cycle. The CPU then initiates an interrupt machine cycle that completes the following processing sequence:

1. Reset (clear to “0”) the global interrupt enable bit in SYM register (DI, SYM.3 = “0”) to disable all subsequent interrupts.
2. Save the program counter and status flags to stack.
3. Jump to the interrupt vector to fetch the service routine’s address.
4. Pass control to the interrupt service routine.

When the interrupt service routine is completed, an Interrupt Return instruction (IRET) occurs. The IRET restores the PC and status flags, and sets SYM.3 to “1” (EI). This allows the CPU to process the next interrupt request.

5.1.7 Generating interrupt Vector Addresses

The interrupt vector area in ROM contains the address of interrupt service routine. Vectored interrupt processing follows this sequence:

1. Push the program counter’s low-byte value to stack.
2. Push the program counter’s high-byte value to stack.
3. Push the FLAGS register values to stack.
4. Fetch the service routine’s high-byte address from the vector address 0000H.
5. Fetch the service routine’s low-byte address from the vector address 0001H.
6. Branch to the service routine specified by the 16-bit vector address.

5.1.8 S3F94C8/F94C4 Interrupt Structure

The S3F94C8/F94C4 microcontroller has four peripheral interrupt sources:

- PWM overflow
- Timer 0 match
- P0.0 external interrupt
- P0.1 external interrupt

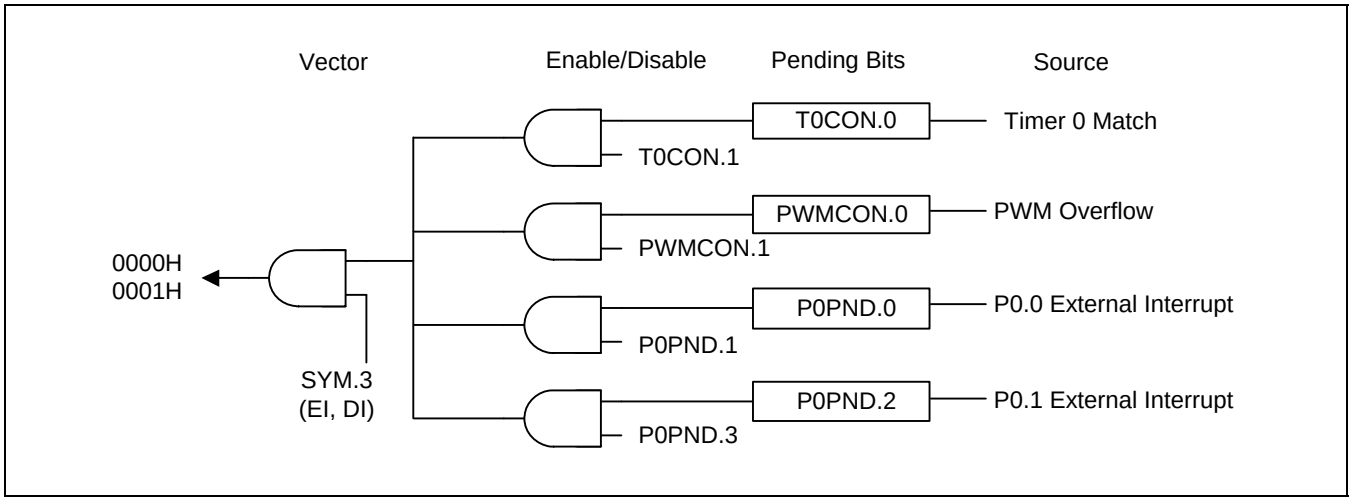


Figure 5-3 S3F94C8/F94C4 Interrupt Structure

5.1.9 Peripheral Interrupt Control Registers

For each interrupt source there is one or more corresponding peripheral control registers that lets you control the interrupt generated by the related peripheral (see *Table 5-1*).

Table 5-1 Interrupt Source Control and Data Registers

Interrupt Source	Register(s)	Register Location(s)
P0.0 external interrupt P0.1 external interrupt	P0CONL P0PND	E7H E8H
Timer 0 match interrupt	T0CON T0DATA	D2H D1H
PWM overflow interrupt	PWMCON PWMDATA PWMDATA1	F3H F2H F0H

6 SAM88RCRI INSTRUCTION SET

6.1 Overview of SAM88RCRI Instruction Set

The SAM88RCRI instruction set is designed to support large register file. It includes full complement of 8-bit arithmetic and logic operations. There are 41 instructions. No special I/O instructions are necessary because I/O control and data registers are mapped directly into the register file. Flexible instructions for bit addressing, rotate, and shift operations complete the powerful data manipulation capabilities of the SAM88RCRI instruction set.

6.1.1 Register Addressing

To access an individual register, an 8-bit address in the range 0–255 or the 4-bit address of a working register is specified. Paired registers can be used to construct 16-bit program memory or data memory addresses. For detailed information about register addressing, refer to *Chapter2 "Address spaces"*.

6.1.2 Addressing Modes

There are six addressing modes, namely, Register (R), Indirect Register (IR), Indexed (X), Direct (DA), Relative (RA), and Immediate (IM). For detailed description of these addressing modes, refer to Chapter 3, “Addressing Modes”.

Table 6-1 Instruction Group Summary

Mnemonic	Operands	Instruction
Load Instructions		
CLR	dst	Clear
LD	dst,src	Load
LDC	dst,src	Load program memory
LDE	dst,src	Load external data memory
LDCD	dst,src	Load program memory and decrement
LDED	dst,src	Load external data memory and decrement
LDCI	dst,src	Load program memory and increment
LDEI	dst,src	Load external data memory and increment
POP	dst	Pop from stack
PUSH	src	Push to stack
Arithmetic Instructions		
ADC	dst,src	Add with carry
ADD	dst,src	Add
CP	dst,src	Compare
DEC	dst	Decrement
INC	dst	Increment
SBC	dst,src	Subtract with carry
SUB	dst,src	Subtract
Logic Instructions		
AND	dst,src	Logical AND
COM	dst	Complement
OR	dst,src	Logical OR
XOR	dst,src	Logical exclusive OR
Program Control Instructions		
CALL	dst	Call procedure
IRET	-	Interrupt return
JP	cc,dst	Jump on condition code
JP	dst	Jump unconditional
JR	cc,dst	Jump relative on condition code
RET	-	Return
Bit Manipulation Instructions		
TCM	dst,src	Test complement under mask
TM	dst,src	Test under mask
Rotate and Shift Instructions		
RL	dst	Rotate left

Mnemonic	Operands	Instruction
RLC	dst	Rotate left through carry
RR	dst	Rotate right
RRC	dst	Rotate right through carry
SRA	dst	Shift right arithmetic
CPU Control Instructions		
CCF	-	Complement carry flag
DI	-	Disable interrupts
EI	-	Enable interrupts
IDLE	-	Enter Idle mode
NOP	-	No operation
RCF	-	Reset carry flag
SCF	-	Set carry flag
STOP	-	Enter stop mode

6.1.3 Flags Register (FLAGS)

The flags register FLAGS contains eight bits that describe the current status of CPU operations. Four of these eight bits, FLAGS.4–FLAGS.7, can be tested and used with conditional jump instructions.

FLAGS register can be set or reset by instructions as long as its outcome does not affect the flags such as Load instruction. Logical and Arithmetic instructions such as AND, OR, XOR, ADD, and SUB can affect the FLAGS register. For example, the AND instruction updates the Zero, Sign, and Overflow flags based on the outcome of the AND instruction. If the AND instruction uses the FLAGS register as the destination, then two write will occur simultaneously to the FLAGS register producing an unpredictable result.

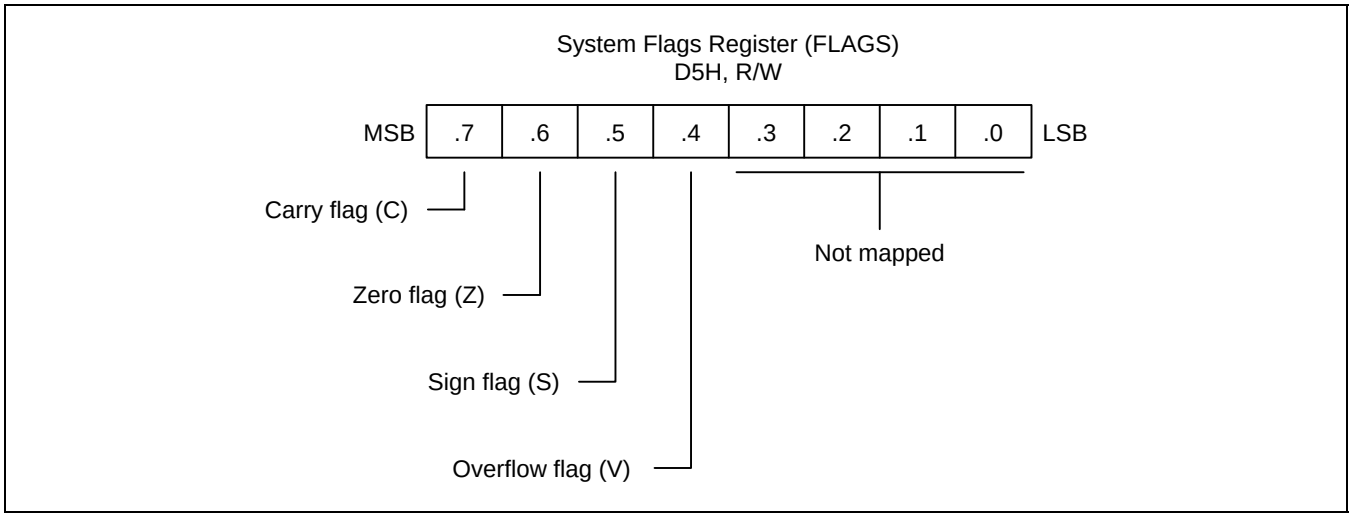


Figure 6-1 System Flags Register (FLAGS)

6.1.4 Flag Descriptions

6.1.4.1 Overflow Flag (FLAGS.4, V)

The V flag is set to “1” when the result of a two’s-complement operation is greater than + 127 or less than – 128. This flag is cleared to “0” following logic operations.

6.1.4.2 Sign Flag (FLAGS.5, S)

The sign bit identifies the state of MSB of the result following arithmetic, logic, rotate, or shift operations. A logic zero indicates a positive number and a logic one indicates a negative number.

6.1.4.3 Zero Flag (FLAGS.6, Z)

For arithmetic and logic operations, the Z flag is set to “1” if the result of the operation is zero. For operations that test register bits, and for shift and rotate operations, the Z flag is set to “1” if the result is logic zero.

6.1.4.4 Carry Flag (FLAGS.7, C)

The C flag is set to “1” if the result from an arithmetic operation generates a carryout from (or a borrow to) the bit 7 position (MSB). After rotate and shift operations, this flag contains the last value shifted out of the specified register. Program instructions can set, clear, or complement the carry flag.

6.1.5 Instruction Set Notation

Table 6-2 Flag Notation Conventions

Flag	Description
C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
0	Cleared to logic zero
1	Set to logic one
*	Set or cleared according to operation
–	Value is unaffected
x	Value is undefined

Table 6-3 Instruction Set Symbols

Symbol	Description
dst	Destination operand
src	Source operand
@	Indirect register address prefix
PC	Program counter
FLAGS	Flags register (D5H)
#	Immediate operand or register address prefix
H	Hexadecimal number suffix
D	Decimal number suffix
B	Binary number suffix
opc	Opcode

Table 6-4 Instruction Notation Conventions

Notation	Description	Actual Operand Range
cc	Condition code	See list of condition codes in Table 6-6.
r	Working register only	Rn (n = 0–15)
rr	Working register pair	RRp (p = 0, 2, 4, ..., 14)
R	Register or working register	reg or Rn (reg = 0–255, n = 0–15)
RR	Register pair or working register pair	reg or RRp (reg = 0–254, even number only, where p = 0, 2, ..., 14)
Ir	Indirect working register only	@Rn (n = 0–15)
IR	Indirect register or indirect working register	@Rn or @reg (reg = 0–255, n = 0–15)
Irr	Indirect working register pair only	@RRp (p = 0, 2, ..., 14)
IRR	Indirect register pair or indirect working register pair	@RRp or @reg (reg = 0–254, even only, where p = 0, 2, ..., 14)
X	Indexed addressing mode	#reg[Rn] (reg = 0–255, n = 0–15)
XS	Indexed (short offset) addressing mode	#addr[RRp] (addr = range – 128 to + 127, where p = 0, 2, ..., 14)
xl	Indexed (long offset) addressing mode	#addr [RRp] (addr = range 0–8191, where p = 0, 2, ..., 14)
da	Direct addressing mode	addr (addr = range 0–8191)
ra	Relative addressing mode	addr (addr = number in the range + 127 to – 128 that is an offset relative to the address of the next instruction)
im	Immediate addressing mode	#data (data = 0–255)

Table 6-5 Opcode Quick Reference

OPCODE MAP									
LOWER NIBBLE (HEX)									
	–	0	1	2	3	4	5	6	7
U	0	DEC R1	DEC IR1	ADD r1,r2	ADD r1,Ir2	ADD R2,R1	ADD IR2,R1	ADD R1,IM	–
P	1	RLC R1	RLC IR1	ADC r1,r2	ADC r1,Ir2	ADC R2,R1	ADC IR2,R1	ADC R1,IM	–
P	2	INC R1	INC IR1	SUB r1,r2	SUB r1,Ir2	SUB R2,R1	SUB IR2,R1	SUB R1,IM	–
E	3	JP IRR1	–	SBC r1,r2	SBC r1,Ir2	SBC R2,R1	SBC IR2,R1	SBC R1,IM	–
R	4	–	–	OR r1,r2	OR r1,Ir2	OR R2,R1	OR IR2,R1	OR R1,IM	–
	5	POP R1	POP IR1	AND r1,r2	AND r1,Ir2	AND R2,R1	AND IR2,R1	AND R1,IM	–
N	6	COM R1	COM IR1	TCM r1,r2	TCM r1,Ir2	TCM R2,R1	TCM IR2,R1	TCM R1,IM	–
I	7	PUSH R2	PUSH IR2	TM r1,r2	TM r1,Ir2	TM R2,R1	TM IR2,R1	TM R1,IM	–
B	8	–	–	–	–	–	–	–	LD r1, x, r2
B	9	RL R1	RL IR1	–	–	–	–	–	LD r2, x, r1
L	A	–	–	CP r1,r2	CP r1,Ir2	CP R2,R1	CP IR2,R1	CP R1,IM	LDC r1, Irr2, xL
E	B	CLR R1	CLR IR1	XOR r1,r2	XOR r1,Ir2	XOR R2,R1	XOR IR2,R1	XOR R1,IM	LDC r2, Irr2, xL
	C	RRC R1	RRC IR1	–	LDC r1,Irr2	–	–	–	LD r1, Ir2
H	D	SRA R1	SRA IR1	–	LDC r2,Irr1	–	–	LD IR1,IM	LD Ir1, r2
E	E	RR R1	RR IR1	LDCD r1,Irr2	LDCI r1,Irr2	LD R2,R1	LD R2,IR1	LD R1,IM	LDC r1, Irr2, xs
X	F	–	–	–	–	CALL IRR1	LD IR2,R1	CALL DA1	LDC r2, Irr1, xs

Table 6-6 Opcode Quick Reference (1)

OPCODE MAP									
LOWER NIBBLE (HEX)									
	–	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
U	0	LD r1,R2	LD r2,R1		JR cc,RA	LD r1,IM	JP cc,DA	INC r1	–
P	1	↓	↓		↓	↓	↓	↓	–
P	2								–
E	3								–
R	4								–
	5								–
N	6								IDLE
I	7	↓	↓		↓	↓	↓	↓	STOP
B	8								DI
B	9								EI
L	A								RET
E	B								IRET
	C								RCF
H	D	↓	↓		↓	↓	↓	↓	SCF
E	E								CCF
X	F	LD r1,R2	LD r2,R1		JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NOP

6.1.6 Condition Codes

The opcode of a conditional jump always contains a 4-bit field called the condition code (cc). This field specifies the conditions under which it has to execute the jump. For example, a conditional jump with the condition code for "equal" after a compare operation only jumps if the two operands are equal. *Table 6-7* lists the condition codes.

The carry (C), zero (Z), sign (S), and overflow (V) flags are used to control the operation of conditional jump instructions.

Table 6-7 Condition Codes

Binary	Mnemonic	Description	Flags Set
0000	F	Always false	–
1000	T	Always true	–
0111 (1)	C	Carry	C = 1
1111 (1)	NC	No carry	C = 0
0110 (1)	Z	Zero	Z = 1
1110 (1)	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110 (1)	EQ	Equal	Z = 1
1110 (1)	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	(Z OR (S XOR V)) = 0
0010	LE	Less than or equal	(Z OR (S XOR V)) = 1
1111 (1)	UGE	Unsigned greater than or equal	C = 0
0111 (1)	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1

NOTE:

1. Indicates the condition codes related to two different mnemonics that also test the same flag. For example, Z and EQ are both true if the zero flag (Z) is set, but after an ADD instruction, Z may be used; after a CP instruction, however, EQ may be used.
2. For operations involving unsigned numbers, the special condition codes UGE, ULT, UGT, and ULE must be used.

6.1.7 Instruction Descriptions

This section contains detailed information and programming examples for each instruction in the SAM87RI instruction set. Information is arranged in a consistent format for improved readability and for fast referencing. The following information is included in each instruction description:

- Instruction name (mnemonic)
- Full instruction name
- Source/destination format of the instruction operand
- Shorthand notation of the instruction's operation
- Textual description of the instruction's effect
- Specific flag settings affected by the instruction
- Detailed description of the instruction's format, execution time, and addressing mode(s)
- Programming example(s) explaining how to use the instruction

6.1.7.1 ADC — Add with Carry

ADC dst,src

Operation: dst ← dst + src + c

The source operand, along with the setting of the carry flag, is added to the destination operand.
Their sum is stored in the destination. The contents of the source remain unaffected. Also, two's-complement addition is performed. In multiple precision arithmetic, this instruction permits the carry from the addition of low-order operands to be carried onto the addition of high-order operands.

Flags:

C: Set if there is a carry from the most significant bit (MSB) of the result; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurs, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	12	r	r
		6	13	r	lr
opc | src | dst	3	6	14	R	R
		6	15	R	IR
opc | dst | src	3	6	16	R	IM

Examples:

Given: R1 = 10H, R2 = 03H, C flag = "1", register 01H = 20H, register 02H = 03H, and register 03H = 0AH:

```
ADC     R1,R2       →   R1 = 14H, R2 = 03H
DC      R1,@R2      →   R1 = 1BH, R2 = 03H
ADC     01H,02H     →   Register 01H = 24H, register 02H = 03H
ADC     01H,@02H    →   Register 01H = 2BH, register 02H = 03H
ADC     01H,#11H    →   Register 01H = 32H
```

In the first example, destination register R1 contains the value 10H, the carry flag is set to "1", and the source working register R2 contains the value 03H. The statement "ADC R1, R2" adds 03H and the carry flag value ("1") to the destination value 10H, leaving 14H in register R1.

6.1.7.2 ADD — Add

ADD dst,src

Operation: dst ← dst + src

The source operand is added to the destination operand and the sum is stored in the destination. The contents of the source remain unaffected. Two's-complement addition is performed.

Flags:
C: Set if there is a carry from the most significant bit (MSB) of the result; cleared therwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if both operands are of the same sign and the result is of the opposite sign; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	02	r	r
		6	03	r	lr
opc | src | dst	3	6	04	R	R
		6	05	R	IR
opc | dst | src	3	6	06	R	IM

Examples:

Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

```
ADD     R1,R2       →   R1 = 15H, R2 = 03H
ADD     R1,@R2      →   R1 = 1CH, R2 = 03H
ADD     01H,02H     →   Register 01H = 24H, register 02H = 03H
ADD     01H,@02H    →   Register 01H = 2BH, register 02H = 03H
ADD     01H,#25H    →   Register 01H = 46H
```

In the first example, destination working register R1 contains 12H and the source working register R2 contains 03H. The statement "ADD R1, R2" adds 03H to 12H, leaving the value 15H in register R1.

6.1.7.3 AND — Logical AND

AND dst, src

Operation: dst ← dst AND src

The source operand is logically ANDed with the destination operand. The result is stored in the destination. The AND operation results in a “1” bit being stored whenever the corresponding bits in the two operands are both logic ones; otherwise a “0” bit value is stored. The contents of the source remain unaffected.

Flags: **C:** Unaffected.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always cleared to “0”.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	52	r	r
				6	53	r	lr
opc	src	dst	3	6	54	R	R
				6	55	R	IR
opc	dst	src	3	6	56	R	IM

Examples:

Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

```
AND     R1,R2       →   R1 = 02H, R2 = 03H
AND     R1,@R2      →   R1 = 02H, R2 = 03H
AND     01H,02H     →   Register 01H = 01H, register 02H = 03H
AND     01H,@02H    →   Register 01H = 00H, register 02H = 03H
AND     01H,#25H    →   Register 01H = 21H
```

In the first example, destination working register R1 contains the value 12H and the source working register R2 contains 03H. The statement “AND R1, R2” logically ANDs the source operand 03H with the destination operand value 12H, leaving the value 02H in register R1.

6.1.7.4 CALL — Call Procedure

CALL dst

Operation:

SP ← SP – 1
@SP ← PCL
SP ← SP –1
@SP ← PCH
PC ← dst

The current contents of the program counter are pushed onto the top of the stack. The program counter value used is the address of the first instruction following the CALL instruction. The specified destination address is then loaded into the program counter and points to the first instruction of a procedure. At the end of the procedure, the return instruction (RET) can be used to return to the original program flow. RET pops the top of the stack back into the program counter.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	3	14	F6	DA
opc | dst	2	12	F4	IRR

Examples:

Given: R0 = 15H, R1 = 21H, PC = 1A47H, and SP = 0B2H:

```
CALL  1521H   →    SP = 0B0H
                   (Memory locations 00H = 1AH, 01H = 4AH, where 4AH is the address that
                         follows the instruction.)
CALL  @RR0    →    SP = 0B0H (00H = 1AH, 01H = 49H)
```

In the first example, if the program counter's value is 1A47H and the stack pointer contains the value 0B2H, the statement "CALL 1521H" pushes the current program counter's value onto the top of the stack. The stack pointer now points to the memory location 00H. The program counter is then loaded with the value 1521H, which specifies the address of the first instruction in the program sequence to be executed.

If the contents of the program counter and stack pointer are the same as in the first example, the statement "CALL @RR0" produces the same result, except that 49H is stored in stack location 01H (because two-byte instruction format was used). The program counter is then loaded with the value 1521H, which specifies the address of the first instruction in the program sequence to be executed.

6.1.7.5 CCF — Complement Carry Flag

CCF

Operation: C ← NOT C

The carry flag (C) is complemented. If C = “1”, the value of the carry flag is changed to logic zero. On the other hand, if C = “0”, the value of the carry flag is changed to logic one.

Flags: **C:** Complemented.
No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	EF

Example:

Given the carry flag = “0”:

```
CCF
```

If the carry flag = “0”, the CCF instruction complements it in the FLAGS register (0D5H), changing its value from logic zero to logic one.

6.1.7.6 CLR — Clear

CLR dst

Operation: dst ← "0"

The destination location is cleared to "0".

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	B0	R
		4	B1	IR

Examples:

Given: Register 00H = 4FH, register 01H = 02H, and register 02H = 5EH:

```
CLR     00H     →    Register 00H = 00H
CLR     @01H    →    Register 01H = 02H, register 02H = 00H
```

In Register (R) addressing mode, the statement "CLR 00H" clears the destination register 00H value to 00H. In the second example, the statement "CLR @01H" uses Indirect Register (IR) addressing mode to clear the 02H register value to 00H.

6.1.7.7 COM — Complement

COM dst

Operation: dst ← NOT dst

The contents of the destination location are complemented (one's complement); all "1s" are changed to "0s", and vice-versa.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to "0".

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	60	R
		4	61	IR

Examples:

Given: R1 = 07H and register 07H = 0F1H:

```
COM     R1      →    R1 = 0F8H
COM     @R1     →    R1 = 07H, register 07H = 0EH
```

In the first example, destination working register R1 contains the value 07H (00000111B). The statement "COM R1" complements all the bits in R1. All logic ones are changed to logic zeros, and vice-versa, leaving the value 0F8H (11111000B).

In the second example, Indirect Register (IR) addressing mode is used to complement the value of destination register 07H (11110001B), leaving the new value 0EH (00001110B).

6.1.7.8 CP — Compare

CP dst,src

Operation: dst – src

The source operand is compared to (subtracted from) the destination operand, and the appropriate flags are set accordingly. The contents of both operands are unaffected by the comparison.

Flags:
C: Set if a "borrow" occurred (src > dst); cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, if the operands were of opposite signs and the sign of the result is the same as the sign of the source operand; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	A2	r	r
		6	A3	r	lr
opc | src | dst	3	6	A4	R	R
		6	A5	R	IR
opc | dst | src	3	6	A6	R	IM

Examples:

```
1.    Given: R1 = 02H and R2 = 03H:
              CP     R1,R2     →  Set the C and S flags
```

Destination working register R1 contains the value 02H and source register R2 contains the value 03H. The statement "CP R1, R2" subtracts the R2 value (source/subtrahend) from the R1 value (destination/minuend). Since a "borrow" occurs and the difference is negative, C and S are "1".

```
2.    Given: R1 = 05H and R2 = 0AH:
              CP     R1,R2
              JP     UGE,SKIP
              INC    R1
              SKIP   LD     R3,R1
```

In this example, destination working register R1 contains the value 05H, which is less than the contents of the source working register R2 (0AH). The statement "CP R1, R2" generates C = "1" and the JP instruction does not jump to the SKIP location. After the statement "LD R3, R1" executes, the value 06H remains in working register R3.

6.1.7.9 DEC — Decrement

DEC dst

Operation: dst ← dst – 1

The contents of the destination operand are decremented by one.

Flags:
C: Unaffected.
Z: Set if the result is “0”; cleared otherwise.
S: Set if result is negative; cleared otherwise.
V: Set if arithmetic overflow occurred, that is, dst value is – 128 (80H) and result value is + 127 (7FH); cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	00	R
		4	01	IR

Examples:

Given: R1 = 03H and register 03H = 10H:

```
DEC   R1      →     R1 = 02H
DEC   @R1     →     Register 03H = 0FH
```

In the first example, if working register R1 contains the value 03H, the statement “DEC R1” decrements the hexadecimal value by one, leaving the value 02H. In the second example, the statement “DEC @R1” decrements the value 10H contained in the destination register 03H by one, leaving the value 0FH.

6.1.7.10 DI — Disable Interrupts

DI

Operation: SYM (3) ← 0

Bit zero of the system mode register, SYM.3, is cleared to “0”, globally disabling all interrupt processing. Interrupt requests will continue to set their respective interrupt pending bits, but the CPU will not service them while interrupt processing is disabled.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	8F

Example:

Given: SYM = 08H:

```
DI
```

If the value of the SYM register is 08H, the statement “DI” leaves the new value 00H in the register and clears SYM.3 to “0”, disabling interrupt processing.

6.1.7.11 EI — Enable Interrupts

EI

Operation: SYM (3) ← 1

An EI instruction sets bit 2 of the system mode register, SYM.3 to “1”. This allows interrupts to be serviced as they occur. If an interrupt’s pending bit was set while interrupt processing was disabled (by executing a DI instruction), it will be serviced when you execute the EI instruction.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	9F

Example:

Given: SYM = 00H:

```
EI
```

If the SYM register contains the value 00H, that is, if interrupts are currently disabled, the statement “EI” sets the SYM register to 08H, enabling all interrupts. (SYM.3 is the enable bit for global interrupt processing.)

6.1.7.12 IDLE — Idle Operation

IDLE

Operation: The IDLE instruction stops the CPU clock while allowing system clock oscillation to continue. Idle mode can be released by an interrupt request (IRQ) or an external reset operation.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	1	4	6F	–	–

Example:

The instructions
```
IDLE
NOP
NOP
NOP
```
Stop the CPU clock but not the system clock.

6.1.7.13 INC — Increment

INC dst

Operation: dst ← dst + 1

The contents of the destination operand are incremented by one.

Flags:
C: Unaffected.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurs, that is dst value is + 127 (7FH) and result is – 128 (80H); cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
dst | opc	1	4	rE r = 0 to F	r
opc | dst	2	4	20	R
		4	21	IR

Examples:

Given: R0 = 1BH, register 00H = 0CH, and register 1BH = 0FH:

```
INC     R0      →     R0 = 1CH
INC     00H     →     Register 00H = 0DH
INC     @R0     →     R0 = 1BH, register 01H = 10H
```

In the first example, if destination working register R0 contains the value 1BH, the statement "INC R0" leaves 1CH value in the same register.

The next example shows the effect of an INC instruction on register 00H. The assumption is INC contains the value 0CH.

In the third example, INC is used in Indirect Register (IR) addressing mode to increment the value of register 1BH from 0FH to 10H.

6.1.7.14 IRET — Interrupt Return

IRET IRET

Operation:

FLAGS	←	@SP
SP	←	SP + 1
PC	←	@SP
SP	←	SP + 2
SYM(2)	←	1

This instruction is used at the end of an interrupt service routine. It restores the flag register and program counter. It also re-enables the global interrupts.

Flags: All flags are restored to their original settings, that is, the settings before the interrupt occurred.

Format:

IRET (Normal)	Bytes	Cycles	Opcode (Hex)
opc	1	10	BF
		12	

6.1.7.15 JP — Jump

JP cc,dst (Conditional)

JP dst (Unconditional)

Operation: If cc is true, PC ← dst

The conditional JUMP instruction transfers program control to the destination address if the condition specified by the condition code (cc) is true; otherwise, the instruction following the JP instruction is executed. The unconditional JP replaces the contents of the PC with the contents of the specified register pair. Then, the control passes to the statement addressed by the PC.

Flags: No flags are affected.

Format: (1)

(2)	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
cc | opc / dst	3	8	ccD cc = 0 to F	DA
opc / dst	2	8	30	IRR

NOTE:

1. 3-byte format is used for a conditional jump and 2-byte format for an unconditional jump.
2. In the first byte of the three-byte instruction format (conditional jump), the condition code and the op code are both four bits.

Examples:

Given: The carry flag (C) = "1", register 00 = 01H, and register 01 = 20H:

```
JP      C,LABEL_W    →    LABEL_W = 1000H, PC = 1000H
JP      @00H         →    PC = 0120H
```

The first example shows a conditional JP. Assuming that the carry flag is set to "1", the statement "JP C, LABEL_W" replaces the contents of the program counter with the value 1000H and transfers control to that location. If the carry flag had not been set, then the control would have passed to the statement immediately following the JP instruction.

The second example shows an unconditional JP. The statement "JP @00" replaces the contents of the program counter with the contents of the register pair 00H and 01H, leaving the value 0120H.

6.1.7.16 JR — Jump Relative

JR cc, dst

Operation: If cc is true, PC ← PC + dst
If the condition specified by the condition code (cc) is true, the relative address is added to the program counter and control passes to the statement whose address is now in the program counter; otherwise, the instruction following the JR instruction is executed (See the list of condition codes).
The range of relative address is + 127 to – 128. The original value of the program counter specifies the address of first instruction byte following the JR statement.

Flags: No flags are affected.

Format:

(NOTE)

cc | opc	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode dst
2	6	ccB	RA
		cc = 0 to F	

NOTE: In the first byte of the two-byte instruction format, the condition code and op code are four bits each.

Example:

Given: The carry flag = “1” and LABEL_X = 1FF7H:

```
JR      C,LABEL_X    →    PC = 1FF7H
```

If the carry flag is set (that is, if the condition code is true), the statement “JR C, LABEL_X” will pass control to the statement whose address is now in the PC. Otherwise, the program instruction following the JR will be executed.

6.1.7.17 LD — Load

LD dst,src

Operation: dst ← src

The contents of source are loaded into the destination. The source contents remain unaffected.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
dst | opc / src	2	4	rC	r	IM
		4	r8	r	R
src | opc / dst	2	4	r9 (r = 0 to F)	R	r
opc / dst | src	2	4	C7	r	lr
		4	D7	lr	r
opc / src / dst	3	6	E4	R	R
		6	E5	R	IR
opc / dst / src	3	6	E6	R	IM
		6	D6	IR	IM
opc / src / dst	3	6	F5	IR	R
opc / dst | src / x	3	6	87	r	x [r]
opc / src | dst / x	3	6	97	x [r]	r

Examples:

Given: R0 = 01H, R1 = 0AH, register 00H = 01H, register 01H = 20H, register 02H = 02H, LOOP = 30H, and register 3AH = 0FFH:

```
LD    R0,#10H        →     R0 = 10H
LD    R0,01H         →     R0 = 20H, register 01H = 20H
LD    01H,R0         →     Register 01H = 01H, R0 = 01H
LD    R1,@R0         →     R1 = 20H, R0 = 01H
LD    @R0,R1         →     R0 = 01H, R1 = 0AH, register 01H = 0AH
LD    00H,01H        →     Register 00H = 20H, register 01H = 20H
LD    02H,@00H       →     Register 02H = 20H, register 00H = 01H
LD    00H,#0AH       →     Register 00H = 0AH
LD    @00H,#10H      →     Register 00H = 01H, register 01H = 10H
LD    @00H,02H       →     Register 00H = 01H, register 01H = 02, register 02H = 02H
LD    R0,#LOOP[R1]   →     R0 = 0FFH, R1 = 0AH
LD    #LOOP[R0],R1   →     Register 31H = 0AH, R0 = 01H, R1 = 0AH
```

6.1.7.18 LDC/LDE — Load Memory

LDC/LDE dst, src

Operation: dst ← src

This instruction loads a byte from program or data memory into a working register, or vice-versa. The source values remain unaffected. LDC and LDE refer to program memory and data memory respectively. The assembler changes “Irr” or “rr” values to an even number for program memory and to an odd number for data memory.

Flags: No flags are affected.

Format:

					Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
1.	opc	dst | src			2	10	C3	r	Irr
2.	opc	src | dst			2	10	D3	Irr	r
3.	opc	dst | src	XS		3	12	E7	r	XS [rr]
4.	opc	src | dst	XS		3	12	F7	XS [rr]	r
5.	opc	dst | src	XLL	XLH	4	14	A7	r	XL [rr]
6.	opc	src | dst	XLL	XLH	4	14	B7	XL [rr]	r
7.	opc	dst | 0000	DAL	DAH	4	14	A7	r	DA
8.	opc	src | 0000	DAL	DAH	4	14	B7	DA	r
9.	opc	dst | 0001	DAL	DAH	4	14	A7	r	DA
10.	opc	src | 0001	DAL	DAH	4	14	B7	DA	r

NOTE:

1. The source (src) or working register pair [rr] for formats 5 and 6 cannot use register pair 0–1.
2. For formats 3 and 4, the destination address “XS [rr]” and source address “XS [rr]” are one byte each.
3. For formats 5 and 6, the destination address “XL [rr]” and the source address “XL [rr]” are two bytes each.
4. The DA and r source values for formats 7 and 8 address program memory; the second set of values for formats 9 and 10 address data memory.

Examples:

Given: R0 = 11H, R1 = 34H, R2 = 01H, R3 = 04H, R4 = 00H, R5 = 60H; Program memory locations 0061 = AAH, 0103H = 4FH, 0104H = 1A, 0105H = 6DH, and 1104H = 88H. External data memory locations 0061H = BBH, 0103H = 5FH, 0104H = 2AH, 0105H = 7DH, and 1104H = 98H:

```
LDC                 R0,@RR2             ; R0 ← contents of program memory
                                          location 0104H
                                        ; R0 = 1AH, R2 = 01H, R3 = 04H

LDE                 R0,@RR2             ; R0 ← contents of external data
                                          memory location 0104H
                                        ; R0 = 2AH, R2 = 01H, R3 = 04H

LDC (NOTE)          @RR2,R0             ; 11H (contents of R0)
                                          is loaded into program memory
                                        ; location 0104H (RR2),
                                        ; working registers R0, R2, R3 → no change

LDE                 @RR2,R0             ; 11H (contents of R0) is
                                          loaded into external data memory
                                        ; location 0104H (RR2),
                                        ; working registers R0, R2, R3 → no change

LDC                 R0,#01H[RR4]        ; R0 ← contents of program memory
                                          location 0061H
                                        ; (01H + RR4),
                                        ; R0 = AAH, R2 = 00H, R3 = 60H

LDE                 R0,#01H[RR4]        ; R0 ← contents of external data
                                          memory location 0061H
                                        ; (01H + RR4), R0 = BBH, R4 = 00H, R5 = 60H

LDC (NOTE)          #01H[RR4],R0        ; 11H (contents of R0) is loaded into
                                          program memory location
                                        ; 0061H (01H + 0060H)

LDE                 #01H[RR4],R0        ; 11H (contents of R0) is loaded into
                                          external data memory
                                        ; location 0061H (01H + 0060H)

LDC                 R0,#1000H[RR2]      ; R0 ← contents of program memory location 1104H
                                        ; (1000H + 0104H), R0 = 88H, R2 = 01H, R3 = 04H

LDE                 R0,#1000H[RR2]      ; R0 ← contents of external data memory
                                          location 1104H
                                        ; (1000H + 0104H), R0 = 98H, R2 = 01H, R3 = 04H
```

```
LDC             R0,1104H        ; R0 ← contents of program memory
                                  location 1104H, R0 = 88H

LDE             R0,1104H        ; R0 ← contents of external data memory
                                  location 1104H,
                                ; R0 = 98H

LDC (NOTE)      1105H,R0        ; 11H (contents of R0) is loaded into
                                  program memory location
                                ; 1105H, (1105H) ← 11H

LDE             1105H,R0        ; 11H (contents of R0) is loaded into
                                  external data memory
                                ; location 1105H, (1105H) ← 11H
```

NOTE: These instructions are not supported by masked ROM type devices.

6.1.7.19 LDCD/LDED — Load Memory and Decrement

LDCD/LDED dst, src

Operation:

dst ← src
rr ← rr – 1

These instructions are used for user stacks or block transfers of data from program or data memory to the register file. A working register pair specifies the address of the memory location. The contents of the source location are loaded into the destination location. The memory address is then decremented. The contents of source remain unaffected.
LDCD and LDED reference program memory and external data memory, respectively. The assembler changes “Irr” to an even number for program memory and to an odd number for data memory.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	10	E2	r	Irr

Examples:

Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory location 1033H = 0CDH, and external data memory location 1033H = 0DDH:

```
LDCD        R8,@RR6      ; 0CDH (contents of program memory location 1033H) is loaded
                         ; into R8 and RR6 is decremented by one
                         ; R8 = 0CDH, R6 = 10H, R7 = 32H (RR6 ← RR6 – 1)

LDED        R8,@RR6      ; 0DDH (contents of data memory location 1033H) is loaded
                         ; into R8 and RR6 is decremented by one (RR6 ← RR6 – 1)
                         ; R8 = 0DDH, R6 = 10H, R7 = 32H
```

6.1.7.20 LDCI/LDEI — Load Memory and Increment

LDCI/LDEI dst,src

Operation:

dst ← src
rr ← rr + 1

These instructions are used for user stacks or block transfers of data from program or data memory to the register file. A working register pair specifies the address of the memory location.
The contents of the source location are loaded into the destination location. The memory address is then incremented automatically. The contents of source remain unaffected. LDCI and LDEI refer to program memory and external data memory, respectively. The assembler changes “Irr” to even for program memory and to odd for data memory.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	10	E3	r	Irr

Examples:

Given: R6 = 10H, R7 = 33H, R8 = 12H, program memory locations 1033H = 0CDH and 1034H = 0C5H; external data memory locations 1033H = 0DDH and 1034H = 0D5H:

```
LDCI        R8,@RR6      ; 0CDH (contents of program memory location 1033H) is loaded
                         ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                         ; R8 = 0CDH, R6 = 10H, R7 = 34H

LDEI        R8,@RR6      ; 0DDH (contents of data memory location 1033H) is loaded
                         ; into R8 and RR6 is incremented by one (RR6 ← RR6 + 1)
                         ; R8 = 0DDH, R6 = 10H, R7 = 34H
```

6.1.7.21 NOP — No Operation

NOP

Operation: When the CPU executes this instruction, no action is performed. Typically, one or more NOPs are executed in sequence to affect a timing delay of variable duration.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	FF

Example:

When the instruction
`NOP`
is encountered in a program, no operation occurs. Instead, there is a delay in instruction execution time.

6.1.7.22 OR — Logical OR

OR dst,src

Operation: dst ← dst OR src

The source operand is logically ORed with destination operand. Its result is stored in destination. The contents of source remain unaffected. The OR operation results in a “1” being stored whenever either of the corresponding bits in the two operands is a “1”; otherwise a “0” is stored.

Flags:
C: Unaffected.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always cleared to “0”.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	42	r	r
		6	43	r	lr
opc | src | dst	3	6	44	R	R
		6	45	R	IR
opc | dst | src	3	6	46	R	IM

Examples:

Given: R0 = 15H, R1 = 2AH, R2 = 01H, register 00H = 08H, register 01H = 37H, and register 08H = 8AH:

```
OR    R0,R1        → R0 = 3FH, R1 = 2AH
OR    R0,@R2       → R0 = 37H, R2 = 01H, register 01H = 37H
OR    00H,01H      → Register 00H = 3FH, register 01H = 37H
OR    01H,@00H     → Register 00H = 08H, register 01H = 0BFH
OR    00H,#02H     → Register 00H = 0AH
```

In the first example, if working register R0 contains the value 15H and register R1 contains the value 2AH, the statement “OR R0, R1” logical-ORs both R0 and R1 register contents and stores the result (3FH) in destination register R0.

The other examples show the use of logical OR instruction with the various addressing modes and formats.

6.1.7.23 POP — Pop From Stack

POP dst

Operation: dst ← @SP
SP ← SP + 1

The contents of location addressed by stack pointer are loaded into the destination. The stack pointer is then incremented by one.

Flags: No flags affected.

Format:

opc	dst

Bytes	Cycles	Opcode (Hex)	Addr Mode dst
2	8	50	R
	8	51	IR

Examples:

Given: Register 00H = 01H, register 01H = 1BH, SP (0D9H) = 0BBH, and stack register 0BBH = 55H:

```
POP   00H     →     Register 00H = 55H, SP = 0BCH
POP   @00H    →     Register 00H = 01H, register 01H = 55H, SP = 0BCH
```

In the first example, general register 00H contains the value 01H. The statement “POP 00H” loads the contents of location 0BBH (55H) into destination register 00H and then increments the stack pointer by one. Register 00H then contains the value 55H and SP points to location 0BCH.

6.1.7.24 PUSH — Push To Stack

PUSH src

Operation:

SP ← SP – 1
@SP ← src

A PUSH instruction decrements the stack pointer value and loads the contents of source (src) into the location addressed by decremented stack pointer. The operation then adds the new value to the top of the stack.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | src	2	8	70	R
		8	71	IR

Examples:

Given: Register 40H = 4FH, register 4FH = 0AAH, SP = 0C0H:

```
PUSH    40H     →   Register 40H = 4FH, stack register 0BFH = 4FH,
                    SP = 0BFH
PUSH    @40H    →   Register 40H = 4FH, register 4FH = 0AAH, stack register
                    0BFH = 0AAH, SP = 0BFH
```

In the first example, if the stack pointer contains the value 0C0H, and general register 40H contains the value 4FH, the statement “PUSH 40H” decrements the stack pointer from 0C0 to 0BFH. It then loads the contents of register 40H into location 0BFH. Register 0BFH then contains the value 4FH and SP points to location 0BFH.

6.1.7.25 RCF — Reset Carry Flag

RCF RCF

Operation: C ← 0

The carry flag is cleared to logic zero, regardless of its previous value.

Flags: **C:** Cleared to “0”.

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	CF

Example:

Given: C = “1” or “0”:
The instruction RCF clears the carry flag (C) to logic zero.

6.1.7.26 RET — Return

RET

Operation:

PC ← @SP
SP ← SP + 2

The RET instruction is normally used to return to the previously executing procedure at the end of a procedure entered by a CALL instruction. The contents of the location addressed by the stack pointer are popped into the program counter. The next statement to be executed is addressed by the new program counter value.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	8	AF
		10	

Example:

Given: SP = 0BCH, (SP) = 101AH, and PC = 1234:

```
RET     →     PC = 101AH, SP = 0BEH
```

The statement “RET” pops the contents of stack pointer location 0BCH (10H) into the high byte of program counter. The stack pointer then pops the value in location 0BDH (1AH) into the program counter’s low byte, following which the instruction at location 101AH is executed. The stack pointer now points to memory location 0BEH.

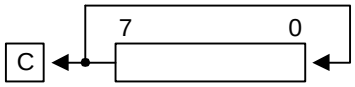
6.1.7.27 RL — Rotate Left

RL dst

Operation:

C ← dst (7)
dst (0) ← dst (7)
dst (n + 1) ← dst (n), n = 0–6

The contents of destination operand are rotated left one bit position. The initial value of bit 7 is moved to the bit zero (LSB) position. Also, this value replaces the carry flag.



Flags:

C: Set if the bit rotated from the most significant bit position (bit 7) was “1”.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurs, that is, if the sign of the destination changed during rotation; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	90	R
		4	91	IR

Examples:

Given: Register 00H = 0AAH, register 01H = 02H and register 02H = 17H:

```
RL    00H     →    Register 00H = 55H, C = “1”
RL    @01H    →    Register 01H = 02H, register 02H = 2EH, C = “0”
```

In the first example, if general register 00H contains the value 0AAH (10101010B), the statement “RL 00H” rotates the 0AAH value left one bit position, leaving the new value 55H (01010101B) and setting the carry and overflow flags.

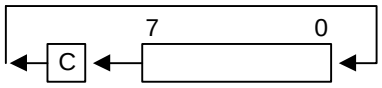
6.1.7.28 RLC — Rotate Left Through Carry

RLC dst

Operation:

dst (0) ← C
C ← dst (7)
dst (n + 1) ← dst (n), n = 0–6

The contents of the destination operand with the carry flag are rotated left one bit position. The initial value of bit 7 replaces the carry flag (C); the initial value of carry flag replaces bit zero.



Flags:

C: Set if the bit rotated from the most significant bit position (bit 7) was “1”.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurs, that is, if the sign of the destination changes during rotation; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	10	R
		4	11	IR

Examples:

Given: Register 00H = 0AAH, register 01H = 02H, and register 02H = 17H, C = “0”:

```
RLC   00H     →    Register 00H = 54H, C = “1”
RLC   @01H    →    Register 01H = 02H, register 02H = 2EH, C  = “0”
```

In the first example, if general register 00H has the value 0AAH (10101010B), the statement “RLC 00H” rotates 0AAH one bit position to the left. The initial value of bit 7 sets the carry flag and the initial value of C flag replaces bit zero of register 00H, leaving the value 55H (01010101B). The MSB of register 00H resets the carry flag to “1” and sets the overflow flag.

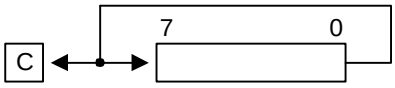
6.1.7.29 RR — Rotate Right

RR dst

Operation:

C ← dst (0)
dst (7) ← dst (0)
dst (n) ← dst (n + 1), n = 0–6

The contents of destination operand are rotated right one bit position. The initial value of bit zero (LSB) is moved to bit 7 (MSB). Also, this value replaces the carry flag (C).



Flags:

C: Set if the bit rotated from the least significant bit position (bit zero) was “1”.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurs, that is, if the sign of destination changed during rotation; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	E0	R
		4	E1	IR

Examples:

Given: Register 00H = 31H, register 01H = 02H, and register 02H = 17H:

```
RR    00H     →    Register 00H =  98H, C = “1”
RR    @01H    →    Register 01H = 02H, register 02H = 8BH, C = “1”
```

In the first example, if general register 00H contains the value 31H (00110001B), the statement “RR 00H” rotates this value one bit position to the right. The initial value of bit zero is moved to bit 7, leaving the new value 98H (10011000B) in the destination register. The initial bit zero also resets the C flag to “1”, and the sign flag and overflow flag are set to “1”.

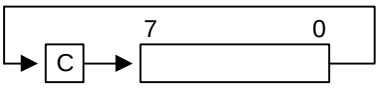
6.1.7.30 RRC — Rotate Right Through Carry

RRC dst

Operation:

dst (7) ← C
C ← dst (0)
dst (n) ← dst (n + 1), n = 0–6

The contents of destination operand and carry flag are rotated right one bit position. The initial value of bit zero (LSB) replaces the carry flag; the initial value of carry flag replaces bit 7 (MSB).



Flags:

C: Set if the bit rotated from the least significant bit position (bit zero) was “1”.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Set if arithmetic overflow occurs, that is, if the sign of destination changes during rotation; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	C0	R
		4	C1	IR

Examples:

Given: Register 00H = 55H, register 01H = 02H, register 02H = 17H, and C = “0”:

```
RRC   00H     →     Register 00H = 2AH, C = “1”
RRC   @01H    →     Register 01H = 02H, register 02H = 0BH, C = “1”
```

In the first example, if general register 00H contains the value 55H (01010101B), the statement “RRC 00H” rotates this value one bit position to the right. The initial value of bit zero (“1”) replaces the carry flag and the initial value of C flag (“1”) replaces bit 7. This leaves the new value 2AH (00101010B) in destination register 00H. Both sign flag and overflow flag are cleared to “0”.

6.1.7.31 SBC — Subtract With Carry

SBC dst,src

Operation: dst ← dst – src – c

The source operand, along with the current value of carry flag, is subtracted from the destination operand and the result is stored in the destination. The contents of source remain unaffected.
Subtraction is performed by adding the two's-complement of source operand to the destination operand. In multiple precision arithmetic, this instruction permits the carry ("borrow") from the subtraction of the low-order operands to be subtracted from the subtraction of high-order operands.

Flags:
C: Set if a borrow occurred (src > dst); cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurs, that is, if the operands were of opposite sign and the sign of result is the same as the sign of source; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	32	r	r
		6	33	r	lr
opc | src | dst	3	6	34	R	R
		6	35	R	IR
opc | dst | src	3	6	36	R	IM

Examples:

Given: R1 = 10H, R2 = 03H, C = "1", register 01H = 20H, register 02H = 03H, and register 03H = 0AH:

```
SBC   R1,R2        →     R1 = 0CH, R2 = 03H
SBC   R1,@R2       →     R1 = 05H, R2 = 03H, register 03H = 0AH
SBC   01H,02H      →     Register 01H = 1CH, register 02H = 03H
SBC   01H,@02H     →     Register 01H = 15H,register 02H = 03H, register 03H = 0AH
SBC   01H,#8AH     →     Register 01H = 95H; C, S, and V = "1"
```

In the first example, if working register R1 contains the value 10H and register R2 the value 03H, the statement "SBC R1,R2" subtracts the source value (03H) and C flag value ("1") from the destination (10H), and then stores the result (0CH) in register R1.

6.1.7.32 SCF — Set Carry Flag

SCF

Operation: C ← 1

The carry flag (C) is set to logic one, regardless of its previous value.

Flags: **C:** Set to “1”.

No other flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)
opc	1	4	DF

Example:

The statement
SCF
Sets the carry flag to logic one.

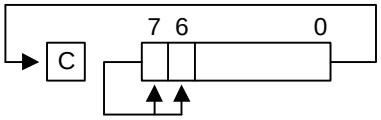
6.1.7.33 SRA — Shift Right Arithmetic

SRA dst

Operation:

dst (7) ← dst (7)
C ← dst (0)
dst (n) ← dst (n + 1), n = 0–6

An arithmetic shift-right of one bit position is performed on the destination operand. Bit zero (LSB) replaces the carry flag. The value of bit 7 (sign bit) remains unchanged and is shifted to bit position 6.



Flags:

C: Set if the bit shifted from the LSB position (bit zero) was “1”.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Always cleared to “0”.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst
opc | dst	2	4	D0	R
		4	D1	IR

Examples:

Given: Register 00H = 9AH, register 02H = 03H, register 03H = 0BCH, and C = “1”:

```
SRA   00H     →    Register 00H = 0CD, C = “0”
SRA   @02H    →    Register 02H = 03H, register 03H = 0DEH, C = “0”
```

In the first example, if general register 00H contains the value 9AH (10011010B), the statement “SRA 00H” shifts the bit values in register 00H to right one bit position. Bit zero (“0”) clears the C flag and bit 7 (“1”) is then shifted to the bit 6 position (bit 7 remains unchanged). This leaves the value 0CDH (11001101B) in destination register 00H.

6.1.7.34 STOP — Stop Operation

STOP

Operation: The STOP instruction stops both the CPU clock and system clock and causes the microcontroller to enter the Stop mode. During Stop mode, the contents of on-chip CPU registers, peripheral registers, and I/O port control and data registers are retained. Stop mode can be released by an external reset operation or external interrupt input. For the reset operation, the RESET pin must be held to Low level until the required oscillation stabilization interval has elapsed.

Flags: No flags are affected.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	src
opc	1	4	7F	–	–

Example:

The statement

```
LD    STOPCON, #0A5H
STOP
NOP
NOP
NOP
```

Halts all microcontroller operations. When STOPCON register does not have #0A5H value and you use STOP instruction, PC is changed to reset address.

6.1.7.35 SUB — Subtract

SUB dst,src

Operation: dst ← dst – src

The source operand is subtracted from destination operand and the result is stored in destination. The contents of the source remain unaffected. Subtraction is performed by adding the two's complement of the source operand to the destination operand.

Flags:
C: Set if a "borrow" occurred; cleared otherwise.
Z: Set if the result is "0"; cleared otherwise.
S: Set if the result is negative; cleared otherwise.
V: Set if arithmetic overflow occurs, that is, if the operands were of opposite signs and the sign of result is the same as sign of the source operand; cleared otherwise.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	22	r	r
		6	23	r	lr
opc | src | dst	3	6	24	R	R
		6	25	R	IR
opc | dst | src	3	6	26	R	IM

Examples:

Given: R1 = 12H, R2 = 03H, register 01H = 21H, register 02H = 03H, register 03H = 0AH:

```
SUB     R1,R2       →   R1 = 0FH, R2 = 03H
SUB     R1,@R2      →   R1 = 08H, R2 = 03H
SUB     01H,02H     →   Register 01H = 1EH, register 02H = 03H
SUB     01H,@02H    →   Register 01H = 17H, register 02H = 03H
SUB     01H,#90H    →   Register 01H = 91H; C, S, and V = "1"
SUB     01H,#65H    →   Register 01H = 0BCH; C and S = "1", V = "0"
```

In the first example, if working register R1 contains the value 12H and if register R2 contains the value 03H, the statement "SUB R1, R2" subtracts the source value (03H) from the destination value (12H) and stores the result (0FH) in destination register R1.

6.1.7.36 TCM — Test Complement Under Mask

TCM dst,src

Operation: (NOT dst) AND src

This instruction tests selected bits in the destination operand for a logic one value. The bits to be tested are specified by setting a “1” bit in the corresponding position of the source operand (mask). The TCM statement complements the destination operand, which is then ANDed with the source mask. The zero (Z) flag can then be checked to determine the result. The destination and source operands remain unaffected.

Flags: **C:** Unaffected.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always cleared to “0”.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	62	r	r
		6	63	r	lr
opc | src | dst	3	6	64	R	R
		6	65	R	IR
opc | dst | src	3	6	66	R	IM

Examples:

Given: R0 = 0C7H, R1 = 02H, R2 = 12H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

```
TCM     R0,R1       →   R0 = 0C7H, R1 = 02H, Z = “1”
TCM     R0,@R1      →   R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = “0”
TCM     00H,01H     →   Register 00H = 2BH, register 01H = 02H, Z = “1”
TCM     00H,@01H    →   Register 00H = 2BH, register 01H = 02H,
                        register 02H = 23H, Z = “1”
TCM     00H,#34     →   Register 00H = 2BH, Z = “0”
```

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 the value 02H (00000010B), the statement “TCM R0, R1” tests bit one in the destination register for a “1” value. Since the mask value corresponds to the test bit, the Z flag is set to logic one and can be tested to determine the result of TCM operation.

6.1.7.37 TM — Test Under Mask

TM dst,src

Operation: dst AND src

This instruction tests selected bits in the destination operand for logic zero value. The bits to be tested are specified by setting a “1” bit in the corresponding position of source operand (mask), which is ANDed with the destination operand. The zero (Z) flag can then be checked to determine the result. The destination and source operands remain unaffected.

Flags: **C:** Unaffected.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to “0”.

Format:

			Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc	dst | src		2	4	72	r	r
				6	73	r	lr
opc	src	dst	3	6	74	R	R
				6	75	R	IR
opc	dst	src	3	6	76	R	IM

Examples:

Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

```
TM      R0,R1       →   R0 = 0C7H, R1 = 02H, Z = “0”
TM      R0,@R1      →   R0 = 0C7H, R1 = 02H, register 02H = 23H, Z = “0”
TM      00H,01H     →   Register 00H = 2BH, register 01H = 02H, Z = “0”
TM      00H,@01H    →   Register 00H = 2BH, register 01H =  02H,
                        register 02H = 23H, Z = “0”
TM      00H,#54H    →   Register 00H = 2BH, Z = “1”
```

In the first example, if working register R0 contains the value 0C7H (11000111B) and register R1 contains the value 02H (00000010B), the statement “TM R0, R1” tests bit one in the destination register for a “0” value. Since the mask value does not match with the test bit, the Z flag is cleared to logic zero and can be tested to determine the result of TM operation.

6.1.7.38 XOR — Logical Exclusive OR

XOR dst,src

Operation: dst ← dst XOR src

The source operand is logically exclusive-ORed with the destination operand, and the result is stored in the destination. The exclusive-OR operation results in a “1” bit being stored whenever the corresponding bits in the operands are different; otherwise, a “0” bit is stored.

Flags: **C:** Unaffected.
Z: Set if the result is “0”; cleared otherwise.
S: Set if the result bit 7 is set; cleared otherwise.
V: Always reset to “0”.

Format:

	Bytes	Cycles	Opcode (Hex)	Addr Mode dst	Addr Mode src
opc | dst | src	2	4	B2	r	r
		6	B3	r	lr
opc | src | dst	3	6	B4	R	R
		6	B5	R	IR
opc | dst | src	3	6	B6	R	IM

Examples:

Given: R0 = 0C7H, R1 = 02H, R2 = 18H, register 00H = 2BH, register 01H = 02H, and register 02H = 23H:

```
XOR   R0,R1        →    R0 = 0C5H, R1 = 02H
XOR   R0,@R1       →    R0 = 0E4H, R1 = 02H, Register 02H = 23H
XOR   00H,01H      →    Register 00H = 29H, Register 01H = 02H
XOR   00H,@01H     →    Register 00H = 08H, Register 01H = 02H, Register 02H = 23H
XOR   00H,#54H     →    Register 00H = 7FH
```

In the first example, if working register R0 contains the value 0C7H and if register R1 contains the value 02H, the statement “XOR R0, R1” logically exclusive-ORs the R1 value with the R0 value and stores the result (0C5H) in the destination register R0

7 CLOCK CIRCUIT

7.1 Overview of Clock Circuit

Using smart option (3FH.1 – .0) in ROM, you can select internal RC oscillator, external RC oscillator, or other external oscillator. If you are using internal oscillator, then normal I/O pins can use XIN (P1.0) and XOUT (P1.1).

An internal RC oscillator source provides a typical 3.2 MHz or 0.5 MHz (in VDD = 5 V) depending on the smart option. On the other hand, an external RC oscillator source provides a typical 4MHz clock for S3F94C8/F94C4.The RC oscillator circuit is supported by an internal capacitor.

An external crystal or ceramic oscillator source provides a 10 MHz (maximum) clock. The XIN and XOUT pins connect the oscillator source to the on-chip clock circuit.

Figure 7-1 and *Figure 7-2* show a simplified external RC oscillator and crystal/ceramic oscillator circuit respectively. When you use an external oscillator, P1.0 and P1.1 must be set to output port to prevent current consumption.

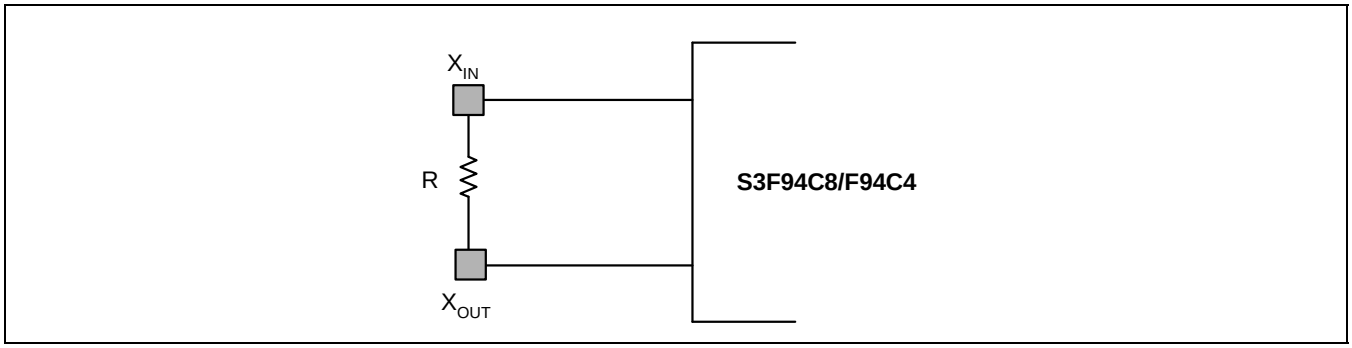


Figure 7-1 Main Oscillator Circuit (RC Oscillator with Internal Capacitor)

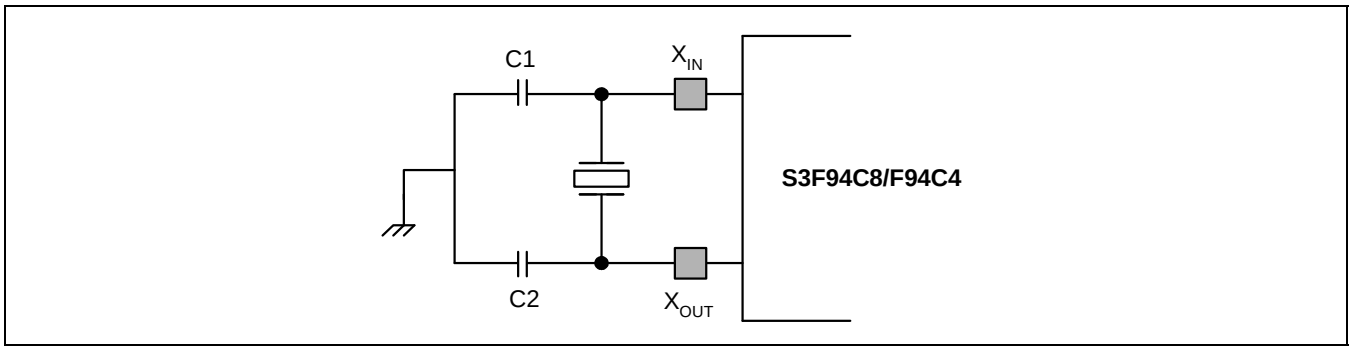


Figure 7-2 Main Oscillator Circuit (Crystal/Ceramic Oscillator)

7.1.1 Main Oscillator Logic

To increase the processing speed and reduce the clock noise, non-divided logic must be implemented for the main oscillator circuit. For the same reasons, high-resolution waveforms (square signal edges) must be generated so that the CPU can process logic operations efficiently.

7.1.2 Clock Status During Power-Down Modes

The two power-down modes, Stop and Idle, affect the clock oscillation as follows:

In Stop mode, the main oscillator “freezes”, that is, it halts the CPU and peripherals. The contents of the register file and current system register values are retained as is. To release the Stop mode and start the oscillator, reset operation must be performed or an external interrupt with RC-delay noise filter (for S3F94C8/F94C4, INT0–INT1) must be applied.

In Idle mode, the internal clock signal is gated off to the CPU, but not to the interrupt control and timer. The current CPU status is preserved, including stack pointer, program counter, and flags. Data in the register file is retained as is. To release the Idle mode, reset operation must be performed or an interrupt (external or internally generated) must be applied.

7.1.3 System Clock Control Register (CLKCON)

The System Clock Control Register, CLKCON, is located in location D4H. This register is read/write addressable and has the following functions:

Oscillator IRQ wake-up function enable/disable (CLKCON.7)

Oscillator frequency divide-by value: non-divided, 2, 8, or 16 (CLKCON.4 and CLKCON.3)

The CLKCON register decides whether an external interrupt can be used to release the Stop mode. (This is called the “IRQ wake-up” function). The IRQ wake-up enable bit is CLKCON.7.

After the reset operation is performed, the external interrupt oscillator wake-up function is enabled, and the fOSC/16 (slowest clock speed) is selected as the CPU clock. If necessary, you can increase the CPU clock speed to fOSC, fOSC /2 or fOSC /8.

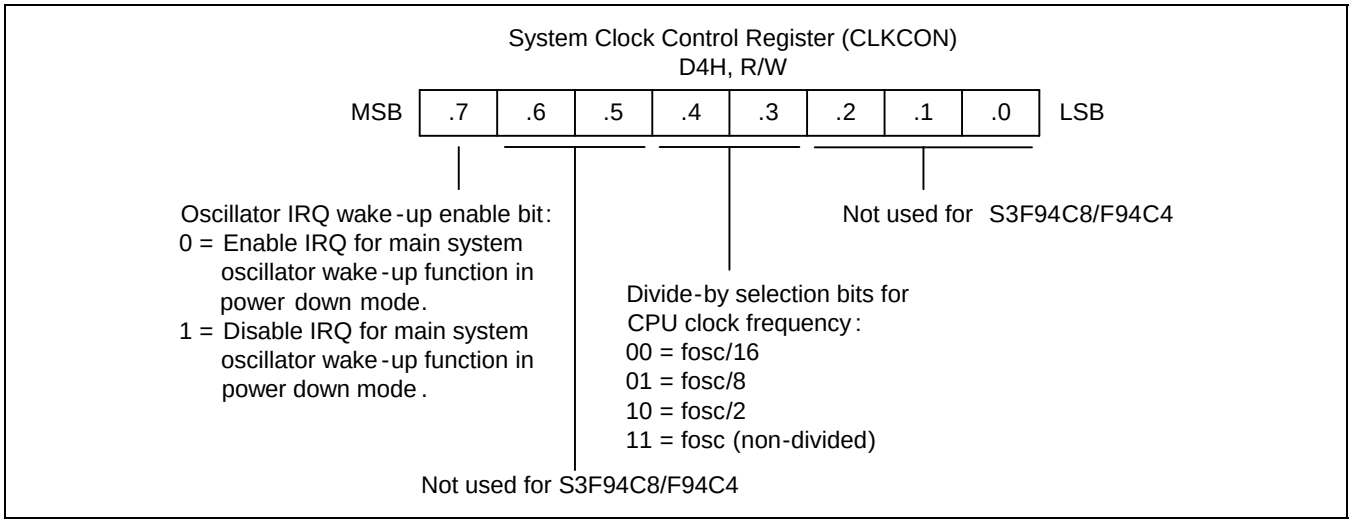
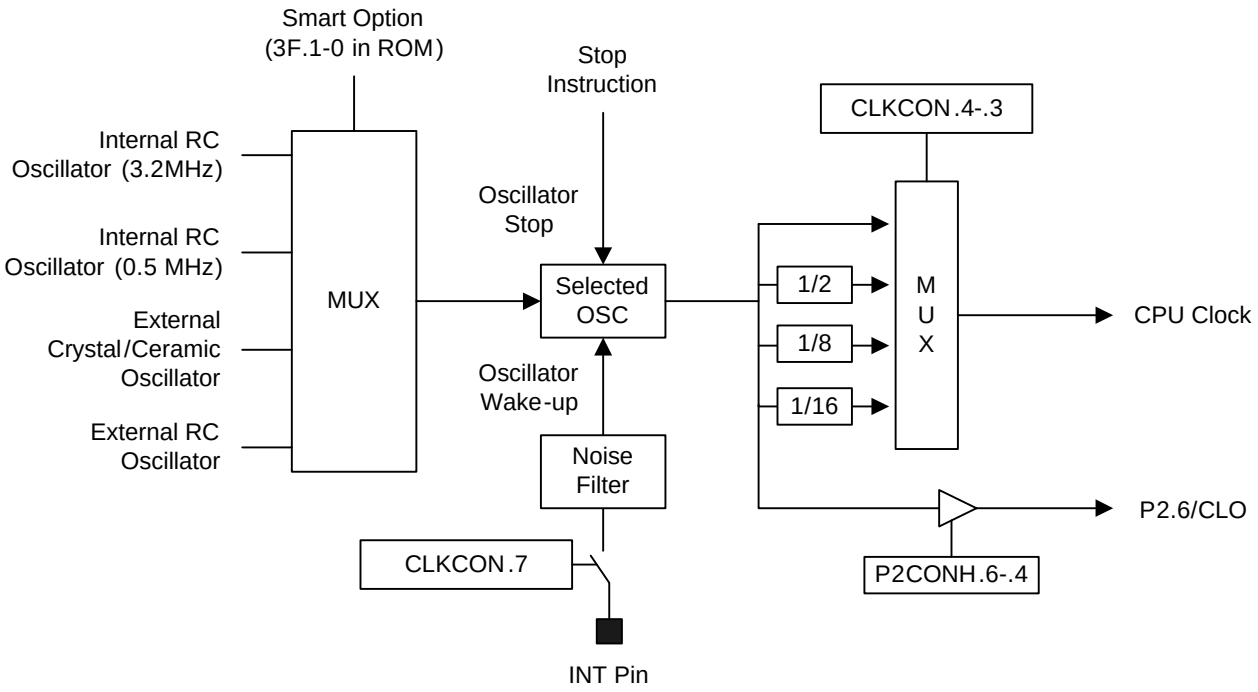


Figure 7-3 System Clock Control Register (CLKCON)



NOTE: An external interrupt (with RC-delay noise filter) can be used to release stop mode and "wake-up" the main oscillator.
In the S3F94C8/F94C4, the INT0-INT1 external interrupts are of this type.

Figure 7-4 System Clock Circuit Diagram

8 RESET AND POWER-DOWN

8.1 System Reset

8.1.1 Overview of System Reset

Use the Smart option (3EH.7) in ROM to select internal RESET (LVR) or external RESET. While using internal RESET (LVR), normal I/O pin can use the nRESET pin (P1.2).

S3F94C8/F94C4 can be RESET in four different ways using:

- External power-on-reset
- External nRESET input pin pulled low
- Digital watchdog peripheral timing out
- Low Voltage Reset (LVR)

During an external power-on reset, the voltage VDD is at High level and the nRESET pin is forced to Low level. The nRESET signal is inputted through a Schmitt trigger circuit where it is synchronized with the CPU clock. This brings the S3F94C8/F94C4 into a known operating status.

To ensure correct start-up, you should not release the nRESET signal before the VDD level is sufficient. This allows MCU to operate at the chosen frequency.

To allow time for internal CPU clock oscillation (so it can stabilize), the nRESET pin must be held to Low level for a minimum time interval after the power supply comes within tolerance. The minimum required oscillation stabilization time for a reset is approximately 52.4 ms (@ 219/fOSC, fOSC = 10 MHz).

When a reset occurs during normal operation (with both VDD and nRESET at High level), the signal at the nRESET pin is forced to Low level and the Reset operation starts. All system and peripheral control registers are then set to their default hardware Reset values (see Table 8-1 for more information).

The MCU provides a watchdog timer function to ensure recovery from software malfunction. If watchdog timer is not refreshed before an end-of-counter condition (overflow) is reached, the internal reset will be activated.

The on-chip Low Voltage Reset allows Static Reset when supply voltage is below a reference value (Typical 1.9, 2.3, 3.0, 3.6, and 3.9 V). Owing to this feature, external reset circuit can be removed. As long as the supply voltage is below the reference value, there is an internal and static RESET. The MCU can only start when the supply voltage rises above the reference value.

While calculating the power consumption, remember that static current of LVR circuit should be added to CPU operating current in any operating mode such as Stop, Idle, and Normal Run when LVR enable in Smart option.

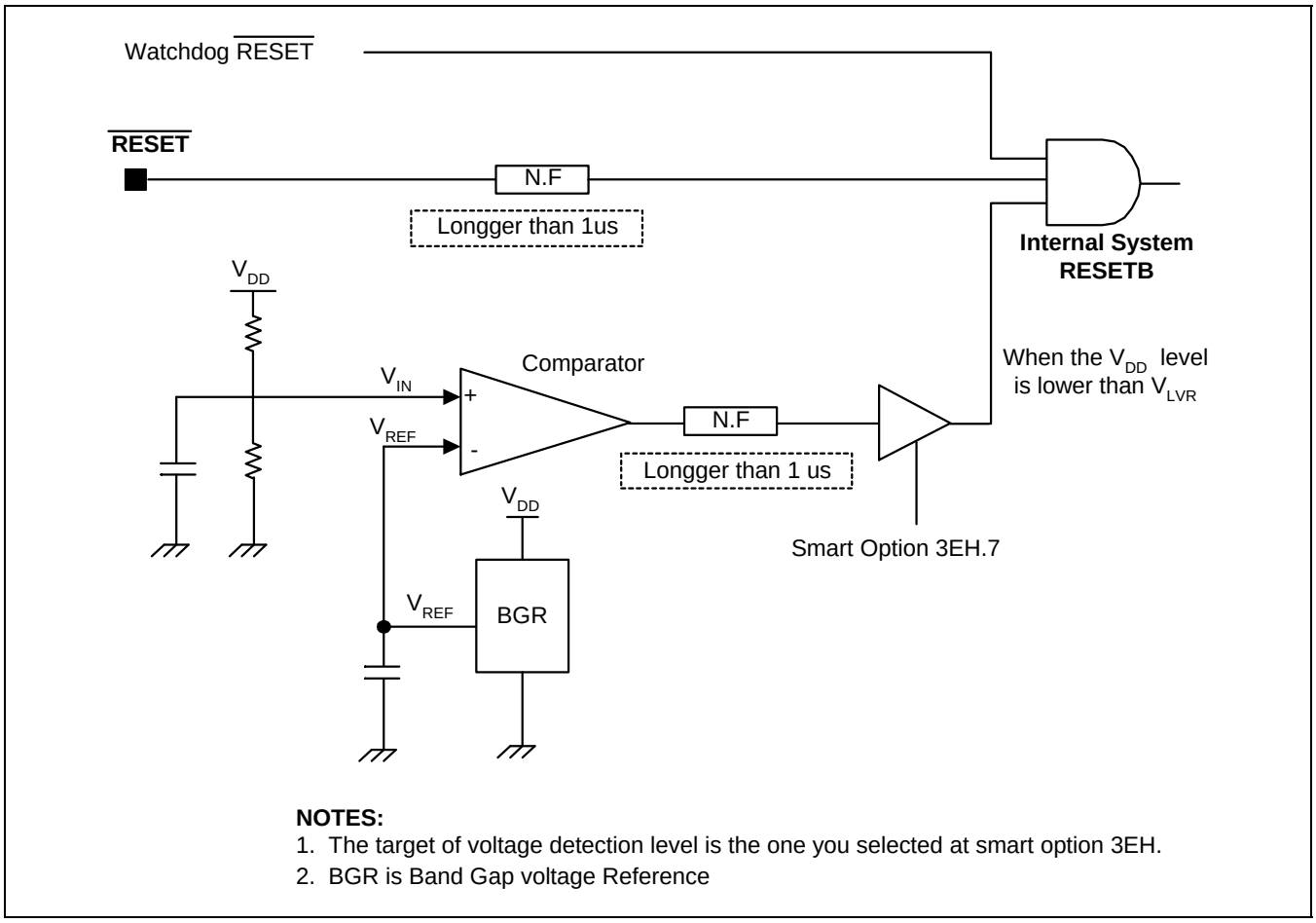
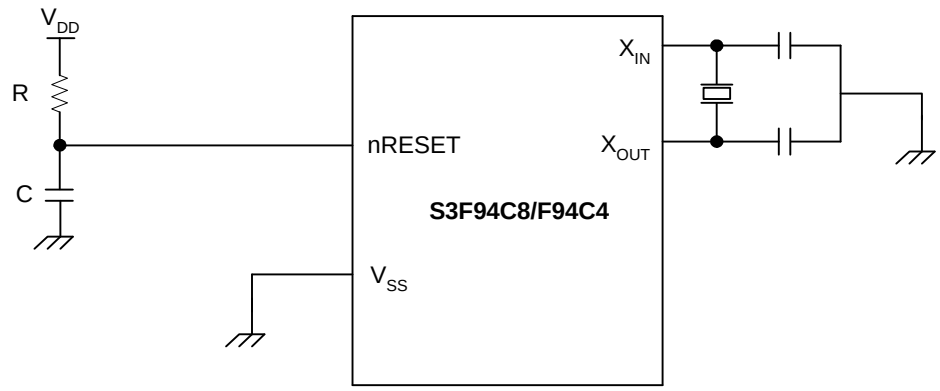


Figure 8-1 Low Voltage Reset Circuit

NOTE: To program the duration of oscillation stabilization interval, configure the appropriate settings in basic timer control register, BTCON, before entering Stop mode. Also, if you do not want to use the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), you can disable it by writing “1010B” to the upper nibble of BTCON.

8.1.1.1 External Reset Pin

When the nRESET pin transits from VIL (low input level of reset pin) to VIH (high input level of reset pin), the reset pulse is generated.



NOTES:
1. R < 100 Kohm is recommended to make sure that the voltage drop across R does not violate the detection of reset pulse.

Figure 8-2 Recommended External RESET Circuit

8.1.1.2 MCU Initialization Sequence

The following sequence of events occurs during a Reset operation:

- All interrupts are disabled.
- The watchdog function (basic timer) is enabled.
- Ports 0–2 are set to input mode
- Peripheral control and data registers are disabled and reset to their initial values (see *Table 8-1* for more information).
- The program counter is loaded with ROM reset address, 0100H.
- When the programmed oscillation stabilization time interval elapses, the address stored in ROM location 0100H (and 0101H) is fetched and executed.

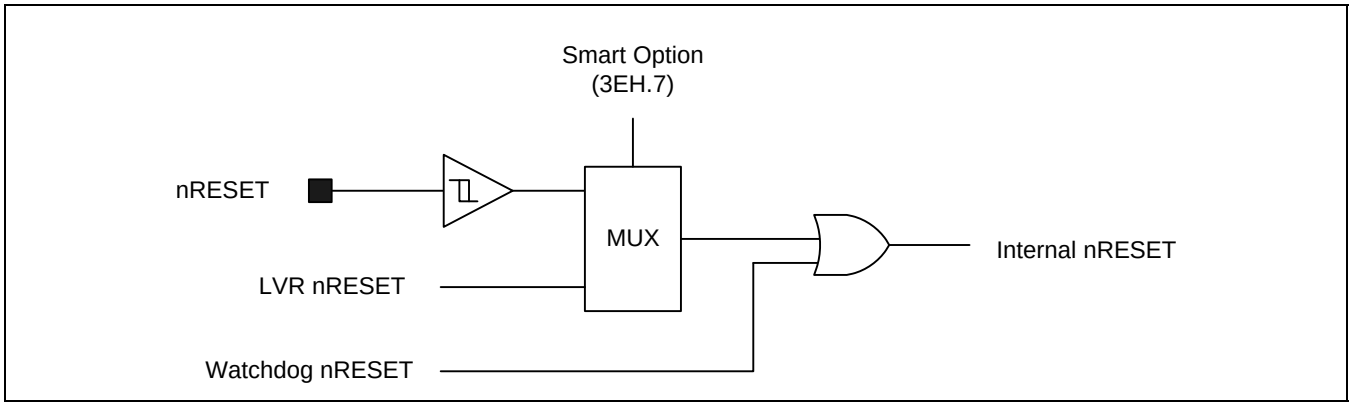


Figure 8-3 Reset Block Diagram

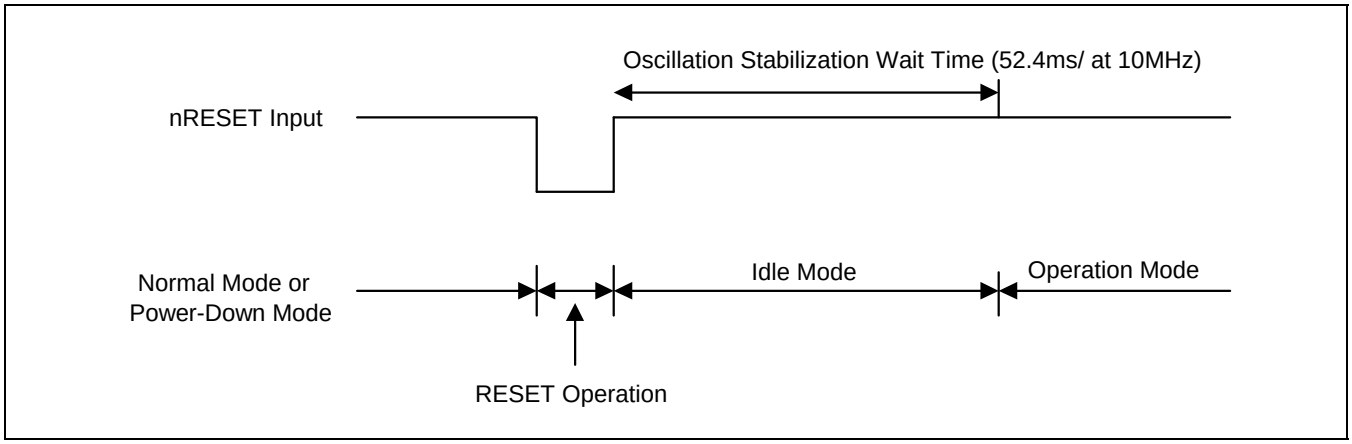


Figure 8-4 Timing for S3F94C8/F94C4 After RESET

8.2 Power-Down Modes

8.2.1 Stop Mode

Stop mode is invoked by the instruction STOP (opcode 7FH). In this mode, the operation of CPU and all peripherals is halted, that is, the on-chip main oscillator stops and the supply current is reduced to less than 4μA (except LVR, which is enabled). All system functions are halted when the clock “freezes”, but data stored in the internal register file is retained. Stop mode can be released by an nRESET signal or an external interrupt.

NOTE: Before executing the STOP instruction, set the STPCON register as “10100101b”.

8.2.1.1 Using RESET to Release Stop Mode

Stop mode is released when nRESET signal is released and returned to High level. All system and peripheral control registers are then reset to their default values and the contents of all data registers are retained. A Reset operation automatically selects a slow clock (fOSC/16) because CLKCON.3 and CLKCON.4 are cleared to “00B”. After the oscillation stabilization interval has elapsed, the CPU executes the system initialization routine by fetching the 16-bit address stored in ROM locations 0100H and 0101H.

8.2.1.2 Using an External Interrupt to Release Stop Mode

External interrupts with an RC-delay noise filter circuit can be used to release Stop mode (Clock-related external interrupts cannot be used). External interrupts INT0-INT1 in S3F94C8/F94C4 interrupt structure meets this criterion.

NOTE: When Stop mode is released by an external interrupt, the current values in system and peripheral control registers are not changed. When you use an interrupt to release Stop mode, the CLKCON.3 and CLKCON.4 register values remain unchanged, and the currently selected clock value is used. If you use an external interrupt for Stop mode release, you can also program the duration of oscillation stabilization interval. To do this, you must put the appropriate value to BTCON register before entering Stop mode.

The external interrupt is serviced when the Stop mode release occurs. Following the IRET from the service routine, the instruction immediately following the one that initiated Stop mode is executed.

8.2.2 Idle Mode

Idle mode is invoked by the instruction IDLE (opcode 6FH). In Idle mode, CPU operations are halted while some peripherals remain active. During this mode, the internal clock signal is gated off to the CPU, but not to the interrupt logic and timer/counters. Port pins retain the same mode (input or output) that they had at the time of entering the Idle mode.

There are two ways to release Idle mode, namely:

1. Execute a Reset. All system and peripheral control registers are reset to their default values and the contents of all data registers are retained. The Reset automatically selects a slow clock (fOSC/16) because CLKCON.3 and CLKCON.4 are cleared to “00B”. If interrupts are masked, Reset is the only way to release Idle mode.

2. Activate any enabled interrupt, causing Idle mode to be released. When you use an interrupt to release Idle mode, the CLKCON.3 and CLKCON.4 register values remain unchanged, and the currently selected clock value is used. The interrupt is then serviced. Following IRET from service routine, the instruction immediately following the one that initiated Idle mode is executed.

NOTE:

1. Only external interrupts that are not clock-related can be used to release the Stop mode. To release Idle mode, however, any type of interrupt (internal or external) can be used.
2. Before entering the STOP or IDLE mode, the ADC must be disabled. Otherwise, the STOP or IDLE current can increase significantly.

8.3 Hardware Reset Values

Table 8-1 lists the values for CPU and system registers, peripheral control registers, and peripheral data registers following a Reset operation in normal operating mode.

- A “1” or a “0” shows the Reset bit value as logic one or logic zero, respectively.
- An “x” means that the bit value is undefined following a reset.
- A dash (“–”) means that the bit is either not used or not mapped.

Table 8-1 Register Values after a Reset

Register Name	Mnemonic	Address and Location		RESET Value (Bit)							
		Address	R/W	7	6	5	4	3	2	1	0
Timer 0 Counter Register	T0CNT	D0H	R	0	0	0	0	0	0	0	0
Timer 0 Data Register	T0DATA	D1H	R/W	1	1	1	1	1	1	1	1
Timer 0 Control Register	T0CON	D2H	R/W	0	0	–	–	0	–	0	0
Location D3H Is Not Mapped											
Clock Control Register	CLKCON	D4H	R/W	0	–	–	0	0	–	–	–
System Flags Register	FLAGS	D5H	R/W	x	x	x	x	–	–	–	–
Locations D6H–D8H Are Not Mapped											
Stack Pointer Register	SP	D9H	R/W	x	x	x	x	x	x	x	x
Location DAH Is Not Mapped											
MDS Special Register	MDSREG	DBH	R/W	0	0	0	0	0	0	0	0
Basic Timer Control Register	BTCON	DCH	R/W	0	0	0	0	0	0	0	0
Basic Timer Counter	BTCNT	DDH	R	0	0	0	0	0	0	0	0
Test Mode Control Register	FTSTCON	DEH	W	–	–	0	0	0	0	0	0
System Mode Register	SYM	DFH	R/W	–	–	–	–	0	0	0	0
Port 0 Data Register	P0	E0H	R/W	0	0	0	0	0	0	0	0
Port 1 Data Register	P1	E1H	R/W	–	–	–	–	–	0	0	0
Port 2 Data Register	P2	E2H	R/W	–	0	0	0	0	0	0	0
Locations E3H–E5H Are Not Mapped											
Port 0 Control Register (High Byte)	P0CONH	E6H	R/W	0	0	0	0	0	0	0	0
Port 0 Control Register	P0CON	E7H	R/W	0	0	0	0	0	0	0	0
Port 0 Interrupt Pending Register	P0PND	E8H	R/W	–	–	–	–	0	0	0	0
Port 1 Control Register	P1CON	E9H	R/W	0	0	–	–	0	0	0	0
Port 2 Control Register (High Byte)	P2CONH	EAH	R/W	–	0	0	0	0	0	0	0
Port 2 Control Register (Low Byte)	P2CONL	EBH	R/W	0	0	0	0	0	0	0	0
Flash Memory Control Register	FMCON	ECH	R/W	0	0	0	0	0	–	–	0

Register Name	Mnemonic	Address and Location		RESET Value (Bit)							
		Address	R/W	7	6	5	4	3	2	1	0
Flash Memory User Programming Enable Register	FMUSR	EDH	R/W	0	0	0	0	0	0	0	0
Flash Memory Sector Address Register (High Byte)	FMSECH	EEH	R/W	0	0	0	0	0	0	0	0
Flash Memory Sector Address Register (Low Byte)	FMSECL	EFH	R/W	0	0	0	0	0	0	0	0
PWM Data Register 1	PWMDATA1	F0H	R/W	0	0	0	0	0	0	0	0
PWM Extension Register	PWMEX	F1H	R/W	0	0	0	0	0	0	0	0
PWM Data Register	PWMDATA	F2H	R/W	0	0	0	0	0	0	0	0
PWM Control Register	PWMCON	F3H	R/W	0	0	–	0	0	0	0	0
STOP Control Register	STOPCON	F4H	R/W	0	0	0	0	0	0	0	0
Locations F5H–F6H Are Not Mapped											
A/D Control Register	ADCON	F7H	R/W	0	0	0	0	0	0	0	0
A/D Converter Data Register (High)	ADDATAH	F8H	R	x	x	x	x	x	x	x	x
A/D Converter Data Register (Low)	ADDATAL	F9H	R	0	0	0	0	0	0	x	x
Locations FAH–FFH Are Not Mapped											

NOTE: –: Not mapped or not used
x: undefined

Example 8-1 Programming Tip — Sample S3F94C8/F94C4 Initialization Routine

```
;--------------<< Interrupt Vector Address >>
            ORG          0000H
            VECTOR 00H,INT_94C4        ; S3F94C8/F94C4 has only one interrupt vector
;--------------<< Smart Option >>
            ORG          003CH
            DB           00H           ; 003CH, must be initialized to 0
            DB           00H           ; 003DH, must be initialized to 0
            DB           0E7H          ; 003EH, enable LVR (2.3 V)
            DB           03H           ; 003FH, internal RC (3.2 MHz in VDD = 5 V )
;--------------<< Initialize System and Peripherals >>
            ORG    0100H
RESET:      DI                         ; Disable Interrupt
            LD     BTCON,#10100011B    ; Disable Watchdog
            LD     CLKCON,#00011000B   ; Select non-divided CPU clock
            LD     SP,#0C0H            ; Stack pointer must be set
            LD     P0CONH,#10101010B   ;
            LD     P0CONL,#10101010B   ; P0.0–P0.7 push-pull output
            LD     P1CON,#00001010B    ; P1.0–P1.1 push-pull output
            LD     P2CONH,#01001010B   ;
            LD     P2CONL,#10101010B   ; P2.0–P2.6 push-pull output
;--------------<< Timer 0 settings >>
            LD     T0DATA,#50H         ; CPU = 3.2 MHz, interrupt interval = 6.4 msec
            LD     T0CON,#01001010B    ; fOSC/256, Timer 0 interrupt enable
;--------------<< Clear all data registers from 00h to 5FH >>
            LD     R0,#0               ; RAM clear
RAM_CLR:    CLR    @R0                 ;
            INC    R0                  ;
            CP     R0,#0BFH            ;
            JP     ULE,RAM_CLR
;--------------<< Initialize other registers >>
            •
            •
            •
            EI                         ; Enable interrupt
```

```
;--------------<< Main loop >>
MAIN:       NOP                         ; Start Main Loop
            LD     BTCON,#02H           ; Enable Watchdog
                                        ; Clear Basic Counter (BTCNT)
            •
            •
            CALL   KEY_SCAN             ;
            •
            •
            •
            CALL   LED_DISPLAY          ;
            •
            •
            •
            CALL   JOB                  ;
            •
            •
            •
            JR     T,MAIN               ;
;--------------<< Subroutines >>
KEY_SCAN:   NOP                         ;
            •
            •
            •
            RET
LED_DISPLAY: NOP                        ;
            •
            •
            •
            RET
JOB:        NOP                         ;
            •
            •
            •
            RET
```

```
;--------------<< Interrupt Service Routines >>
                                    ; Interrupt enable bit and pending bit check
INT_94C4:   TM     T0CON,#00000010B    ; Timer0 interrupt enable check
            JR     Z,NEXT_CHK1         ;
            TM     T0CON,#00000001B    ; If timer0 interrupt occurs,
            JP     NZ,INT_TIMER0       ; T0CON.0 bit will be set.
NEXT_CHK1:
            TM     PWMCOM,#00000010B   ; Check PWM overflow interrupt enable
            JR     Z,NEXT_CHK2         ;
            TM     P0PND,#00000001B    ;
            JP     NZ,PWMOVF_INT       ;
NEXT_CHK2:
            TM     P0PND,#00000010B    ; Check INT0 interrupt enable
            JR     Z,NEXT_CHK3         ;
            TM     P0PND,#00000001B    ;
            JP     NZ,INT0_INT         ;
NEXT_CHK3:
            TM     P0PND,#00001000B    ; Check INT1 interrupt enable
            JP     Z,END_INT           ;
            TM     P0PND,#00000100B    ;
            JP     NZ,INT1_INT         ;
            IRET                       ; Interrupt return
END_INT     ; IRET
;--------------< Timer0 interrupt service routine >
INT_TIMER0:
            •                          ;
            •
            AND    T0CON,#11110110B    ; Clear Pending bit
            IRET                       ; Interrupt return
;--------------< PWM overflow interrupt service routine >
PWMOVF_INT:
            •
            •
            AND    PWMCON,#11110110B   ; Clear Pending bit
            IRET                       ; Interrupt return
;--------------< External interrupt0 service routine >
INT0_INT:
            •
            •
            AND    P0PND,#11111110B    ; Clear INT0 Pending bit
            IRET                       ; Interrupt return
```

```
;--------------< External interrupt1 service routine >
INT1_INT:
            •
            •
            AND    P0PND,#11111011B    ; Clear INT1 Pending bit
            IRET                       ; Interrupt return
            •
            •
            END                        ;
```

9 I/O PORTS

9.1 Overview of I/O Ports

The S3F94C8/F94C4 has three I/O ports with 18 pins in total. You can access these ports directly by writing or reading port data register addresses.

All ports can be configured as LED drive. (High current output: typical 10 mA).

Table 9-1 Overview of S3F94C8/F94C4 Port Configuration

Port	Function Description	Programmability
0	Bit-programmable I/O port for Schmitt trigger input or push-pull output. Pull-up resistors are assigned by software. Port 0 pins can also be used for alternative function. (ADC input, external interrupt input).	Bit
1	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up or pull-down resistors are assignable by software. Port 1 pins can also be used as oscillator input/output or reset input by smart option. P1.2 is input only.	Bit
2	Bit-programmable I/O port for Schmitt trigger input or push-pull, open-drain output. Pull-up resistor are assignable by software. Port 2 can also be used as alternative function (ADC input, CLO, T0 clock output).	Bit

9.1.1 Port Data Registers

Table 9-2 provides an overview of the port data register names, locations, and addressing characteristics. *Figure 9-1* shows the structure of data registers for ports 0-2.

Table 9-2 Port Data Register Summary

Register Name	Mnemonic	Hex	R/W
Port 0 data register	P0	E0H	R/W
Port 1 data register	P1	E1H	R/W
Port 2 data register	P2	E2H	R/W

NOTE: A reset operation clears the P0–P2 data register to “00H”.

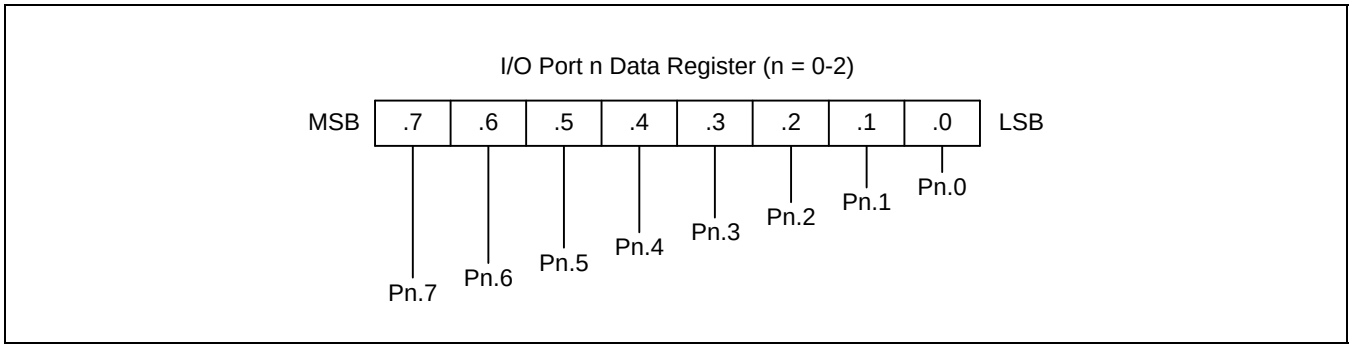


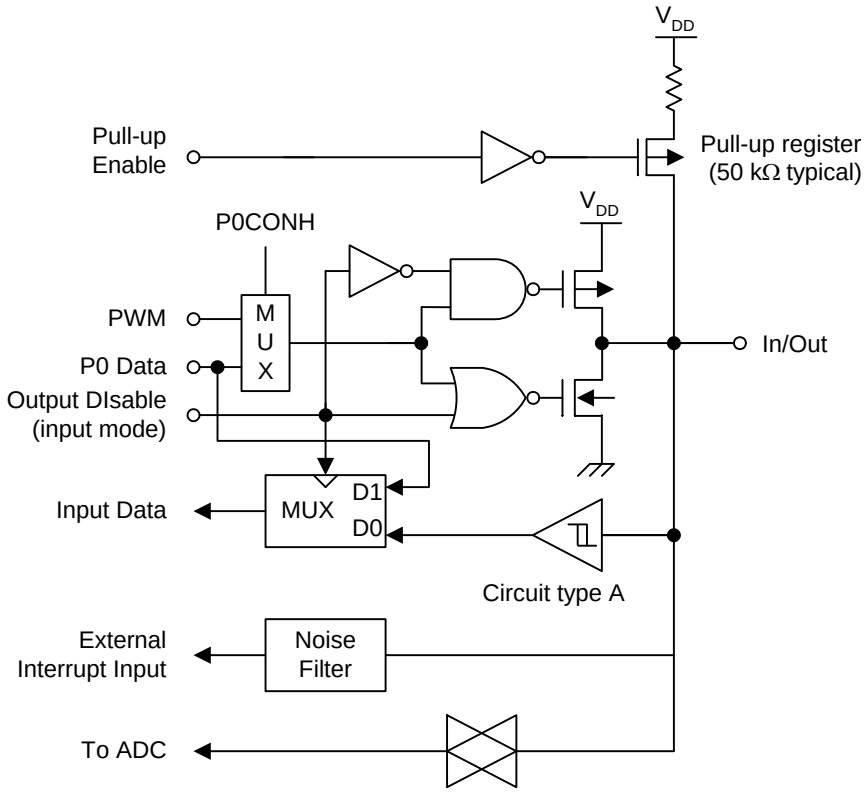
Figure 9-1 Port Data Register Format

9.1.2 Port 0

Port 0 is a bit-programmable, general-purpose I/O port. You can select normal input or push-pull output mode. In addition, you can configure a pull-up resistor to individual pins using control register settings. It is designed for high-current functions such as LED direct drive. Port 0 pins can also be used for alternative functions (ADC input, external interrupt input, and PWM output).

Two control resisters are used to control Port 0, namely, P0CONH (E6H) and P0CONL (E7H).

You access port 0 directly by writing or reading the corresponding port data register, P0 (E0H).



NOTE: I/O pins have protection diodes through V_{DD} and V_{SS}.

Mode	Input Data
Output	D0
Input	D1

Figure 9-2 Port 0 Circuit Diagram

Port 0 Control Register (High Byte)
E6H, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

[.7-.6] Port, P0.7/ADC7 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = A/D converter input (ADC7); schmitt trigger input off

[.5-.4] Port 0, P0.6/ADC6/PWM Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Alternative function (PWM output)
1 0 = Push-pull output
1 1 = A/D converter input (ADC6); schmitt trigger input off

[.3-.2] Port 0, P0.5/ADC5 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = A/D converter input (ADC5); schmitt trigger input off

[.1-.0] Port 0, P0.4/ADC4 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = A/D converter input (ADC4); schmitt trigger input off

Figure 9-3 Port 0 Control Register (P0CONH, High Byte)

Port 0 Control Register (Low Byte)
E7H, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

[.7-.6] Port 0, P0.3/ADC3 Configuration Bits
0 0 = Schmitt trigger input
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = A/D converter input (ADC3); schmitt trigger input off

[.5-.4] Port 0, P0.2/ADC2 Configuration Bits
0 0 = Schmitt trigger input
0 1 = Schmitt trigger input; pull-up enable
1 0 = Push-pull output
1 1 = A/D converter input (ADC2); schmitt trigger input off

[.3-.2] Port 0, P0.1/ADC1/INT1 Configuration Bits
0 0 = Schmitt trigger input/falling edge interrupt input
0 1 = Schmitt trigger input; pull-up enable/falling edge interrupt input
1 0 = Push-pull output
1 1 = A/D converter input (ADC1); schmitt trigger input off

[.1-.0] Port 0, P0.0/ADC0/INT0 Configuration Bits
0 0 = Schmitt trigger input/falling edge interrupt input
0 1 = Schmitt trigger input; pull-up enable/falling edge interrupt input
1 0 = Push-pull output
1 1 = A/D converter input (ADC0); schmitt trigger input off

Figure 9-4 Port 0 Control Register (P0CONL, Low Byte)

Port 0 Interrupt Pending Register
E8H, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

[.7-.4] Not used for S3F94C8/F94CC

[.3] Port 0.1/ADC1/INT1, Interrupt Enable Bit
0 = INT1 falling edge interrupt disable
1 = INT1 falling edge interrupt enable

[.2] Port 0.1/ADC1/INT1, Interrupt Pending Bit
0 = No interrupt pending (when read)
0 = Pending bit clear (when write)
1 = Interrupt is pending (when read)
1 = No effect (when write)

[.1] Port 0.0/ADC0/INT0, Interrupt Enable Bit
0 = INT0 falling edge interrupt disable
1 = INT0 falling edge interrupt enable

[.0] Port 0.0/ADC0/INT0, Interrupt Pending Bit
0 = No interrupt pending (when read)
0 = Pending bit clear (when write)
1 = Interrupt is pending (when read)
1 = No effect (when write)

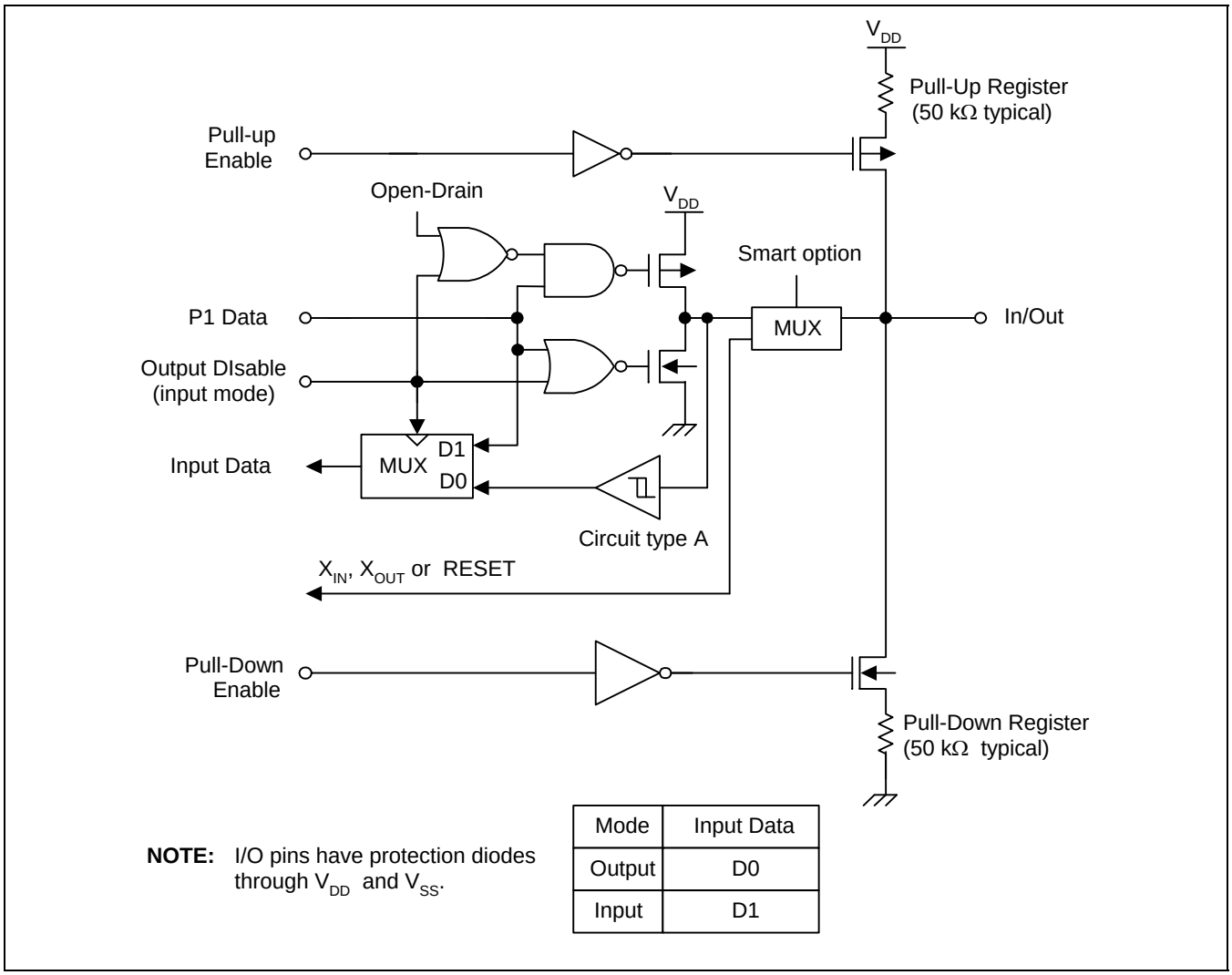
Figure 9-5 Port 0 Interrupt Pending Registers (P0PND)

9.1.3 Port 1

Port 1 is a 3-bit I/O port with individually configurable pins. It can be used for general I/O port (Schmitt trigger input mode, push-pull output mode, or n-channel open-drain output mode). In addition, you can configure a pull-up and pull-down resistor to individual pin using control register settings. It is designed for high-current functions such as LED direct drive. P1.0 and P1.1 are used for oscillator input/output by smart option. On the other hand, P1.2 is used for RESET pin by smart option (LVR disable).

NOTE: When P1.2 is configured as a general I/O port, it can be used only for Schmitt trigger input. P1.2 is also shared with VPP pin for Flash Programming, so it have intrinsic internal pull-down resistor (about 300Kohm), Consider the pull-down resistor when it used as I/O port.

One control register is used to control port 1: P1CON (E9H).You can address port 1 bit directly by writing or reading the port 1 data register, P1 (E1H). When you use external oscillator, P1.0 and P1.1 must be set to output port to prevent current consumption.



NOTE: I/O pins have protection diodes through V_{DD} and V_{SS}.

Mode	Input Data
Output	D0
Input	D1

Figure 9-6 Port 1 Circuit Diagram

Port 1 Control Register
E9H, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

[.7] Port 1.1 N-Channel Open-Drain Enable Bit
0 = Configure P1.1 as a push-pull output
1 = Configure P1.1 as a n-channel open-drain output

[.6] Port 1.0 N-Channel Open-Drain Enable Bit
0 = Configure P1.0 as a push-pull output
1 = Configure P1.0 as a N-channel open-drain output

[.5-.4] Not used for S3F94C8/F94C4

[.3-.2] Port 1, P1.1 Configuration Bits
0 0 = Schmitt trigger input ;
0 1 = Schmitt trigger input ; pull-up enable
1 0 = Push-pull output
1 1 = Schmitt trigger input ; pull-down enable

[.1-.0] Port 1, P1.0 Configuration Bits
0 0 = Schmitt trigger input ;
0 1 = Schmitt trigger input ; pull-up enable
1 0 = Push-pull output
1 1 = Schmitt trigger input ; pull-down enable

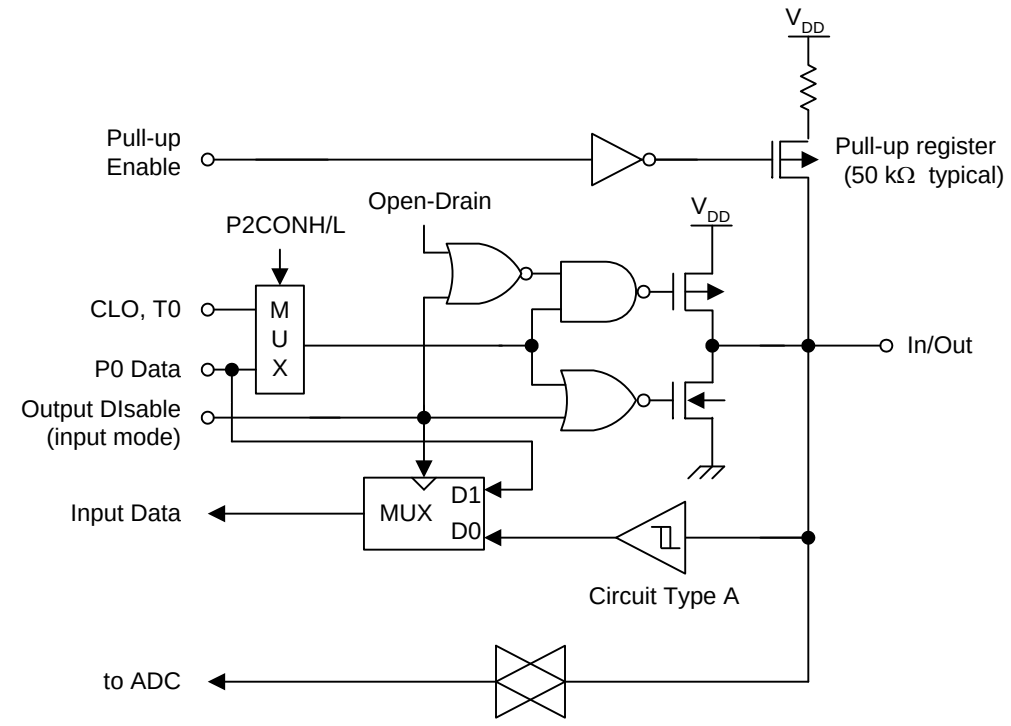
NOTE: 1. When you use external oscillator,P1.0, P1.1 must be set to output port to prevent current consumption.
2. When you enable LVR in smart option, P1.2 (nRESET/VPP) can be and can only be used as input port.

Figure 9-7 Port 1 Control Register (P1CON)

9.1.4 Port 2

Port 2 is a 7-bit I/O port with individually configurable pins. It can be used for general I/O port (Schmitt trigger input mode, push-pull output mode, or N-channel open-drain output mode). You can also use some pins of port 2 ADC input, CLO output, and T0 clock output. In addition, you can configure a pull-up resistor to individual pins using control register settings. It is designed for high-current functions such as LED direct drive.

You can address port 2 bits directly by writing or reading the port 2 data register, P2 (E2H). The port 2 control register, P2CONH and P2CONL, is located at addresses EAH and EBH respectively.



NOTE: I/O pins have protection diodes through V_{DD} and V_{SS}.

Mode	Input Data
Output	D0
Input	D1

Figure 9-8 Port 2 Circuit Diagram

Port 2 Control Register (High Byte)
EAH, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

[.7] Not sued for S3F94C8/F94C4

[.6-.4] Port 2, P2.6/ADC8/ CLO Configuration Bits
0 0 0 = Schmitt trigger input; pull-up enable
0 0 1 = Schmitt trigger input
0 1 x = ADC input
1 0 0 = Push-pull output
1 0 1 = Open-drain output; pull-up enable
1 1 0 = Open-drain output
1 1 1 = Alternative function; CLO output

[.3-.2] Port 2, P2.5 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.1-.0] Port 2, P2.4 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

NOTE: When noise problem is important issue, you had better not use CLO output

Figure 9-9 Port 2 Control Register (P2CONH, High Byte)

Port 2 Control Register (Low Byte)
EBH, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

[.7-.6] Port 2, P2.3 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.5-.4] Port 2, P2.2 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.3-.2] Port 2, P2.1 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = Open-drain output

[.1-.0] Port 2, P2.0 Configuration Bits
0 0 = Schmitt trigger input; pull-up enable
0 1 = Schmitt trigger input
1 0 = Push-pull output
1 1 = T0 match output

Figure 9-10 Port 2 Control Register (P2CONL, Low Byte)

10 BASIC TIMER AND TIMER 0

10.1 Overview of Basic Timer and Timer 0

S3F94C8/F94C4 has two default timers:

- An 8-bit basic timer
- An 8-bit general-purpose timer/counter called timer 0.

10.1.1 Basic Timer

You can use the basic timer (BT) in two different ways:

- As a watchdog timer: To provide an automatic Reset mechanism in the event of system malfunction.
- To signal the end of required oscillation stabilization interval after a Reset or Stop mode release.

The functional components of basic timer block are:

- Clock frequency divider (fOSC divided by 4096, 1024, or 128) with multiplexer
- 8-bit basic timer counter, BTCNT (DDH, read-only)
- Basic timer control register, BTCON (DCH, read/write)

10.1.2 Timer 0

Timer 0 has the following functional components:

- Clock frequency divider (fOSC divided by 4096, 256, 8, or fOSC) with multiplexer
- 8-bit counter (T0CNT), 8-bit comparator, and 8-bit data register (T0DATA)
- Timer 0 control register (T0CON)

10.2 Basic Timer (BT)

10.2.1 Basic Timer Control Register (BTCON)

The basic timer control register, BTCON, selects the input clock frequency to clear the basic timer counter and frequency dividers and to enable or disable the watchdog timer function.

A Reset clears BTCON to “00H”. This enables the watchdog function and selects a basic timer clock frequency of fOSC/4096. To disable the watchdog function, you must write the signature code “1010B” to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit basic timer counter, BTCNT, can be cleared during normal operation by writing a “1” to BTCON.1. To clear the frequency dividers for both the basic timer input clock and timer 0 clock, write “1” to BTCON.0.

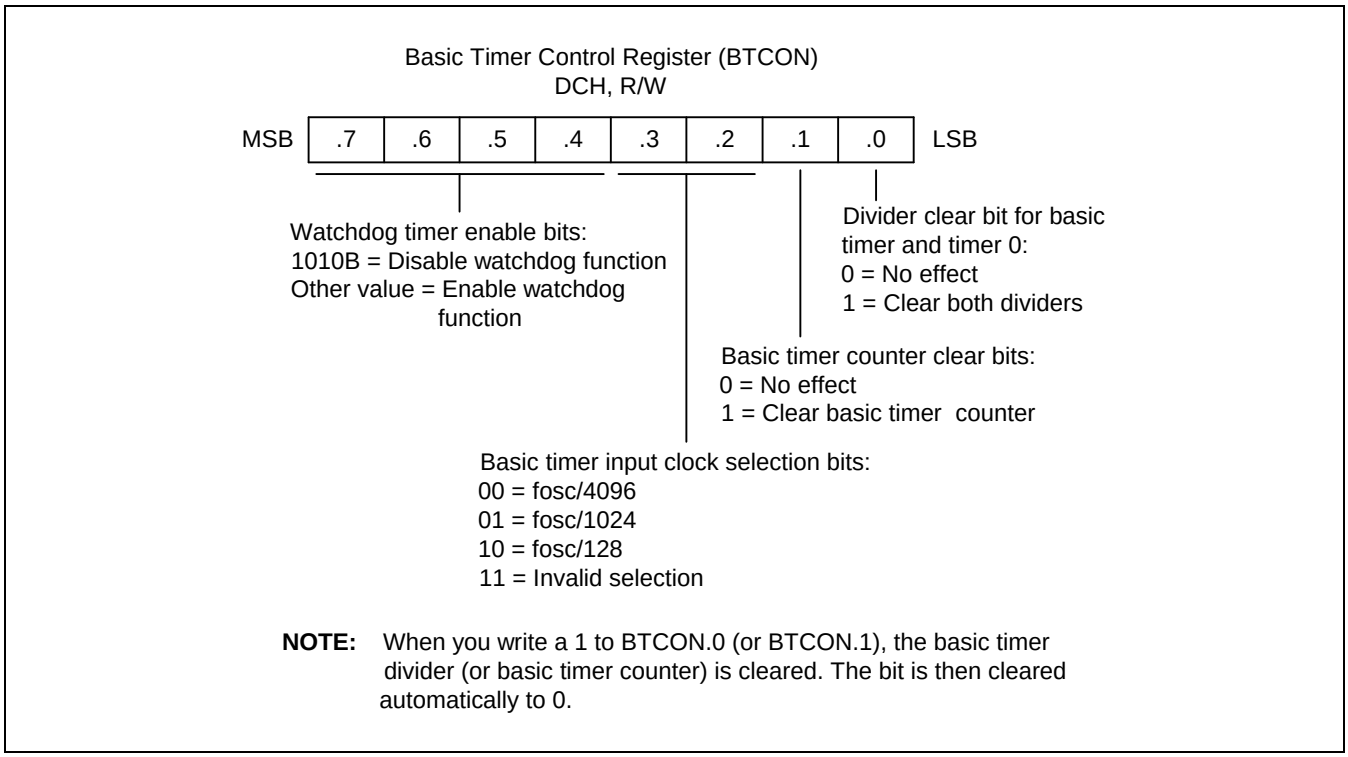

Basic Timer Control Register (BTCON)
DCH, R/W
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Watchdog timer enable bits:
1010B = Disable watchdog function
Other value = Enable watchdog function
Divider clear bit for basic timer and timer 0:
0 = No effect
1 = Clear both dividers
Basic timer counter clear bits:
0 = No effect
1 = Clear basic timer counter
Basic timer input clock selection bits:
00 = fosc/4096
01 = fosc/1024
10 = fosc/128
11 = Invalid selection
NOTE: When you write a 1 to BTCON.0 (or BTCON.1), the basic timer divider (or basic timer counter) is cleared. The bit is then cleared automatically to 0.

Figure 10-1 Basic Timer Control Register (BTCON)

10.2.2 Basic Timer Function

10.2.2.1 Watchdog Timer Function

You can program the basic timer overflow signal (BTOVF) to generate a Reset by setting BTCON.7–BTCON.4 to any value other than “1010B”. (The “1010B” value disables the watchdog function). A Reset clears BTCON to “00H”, enabling the watchdog timer function automatically. A Reset also selects the oscillator clock divided by 4096 as the BT clock.

A Reset is generated whenever a basic timer counter overflow occurs. During normal operation, the application program must prevent the overflow and accompanying reset operation from occurring. To do this, the BTCNT value must be cleared (by writing a “1” to BTCON.1) at regular intervals.

If a system malfunction occurs due to circuit noise or some other error condition, the BT counter clear operation will not be executed and a basic timer overflow will occur, initiating a Reset. In other words, during normal operation, the basic timer overflow loop (a bit 7 overflow of the 8-bit basic timer counter, BTCNT) is always broken by a BTCNT clear instruction. If a malfunction occurs, a Reset is triggered automatically.

10.2.2.2 Oscillation Stabilization Interval Timer Function

You can use the basic timer to program a specific oscillation stabilization interval following a Reset or when the Stop mode is released by an external interrupt.

In Stop mode, whenever a Reset or an external interrupt occurs, the oscillator will start. The BTCNT value then starts increasing at the rate of fOSC/4096 (for Reset) or at the rate of preset clock source (for an external interrupt). When BTCNT.7 is set, a signal is generated to indicate that the stabilization interval has elapsed and to gate the clock signal off to the CPU, so that it can resume normal operation.

The following events occur when Stop mode is released:

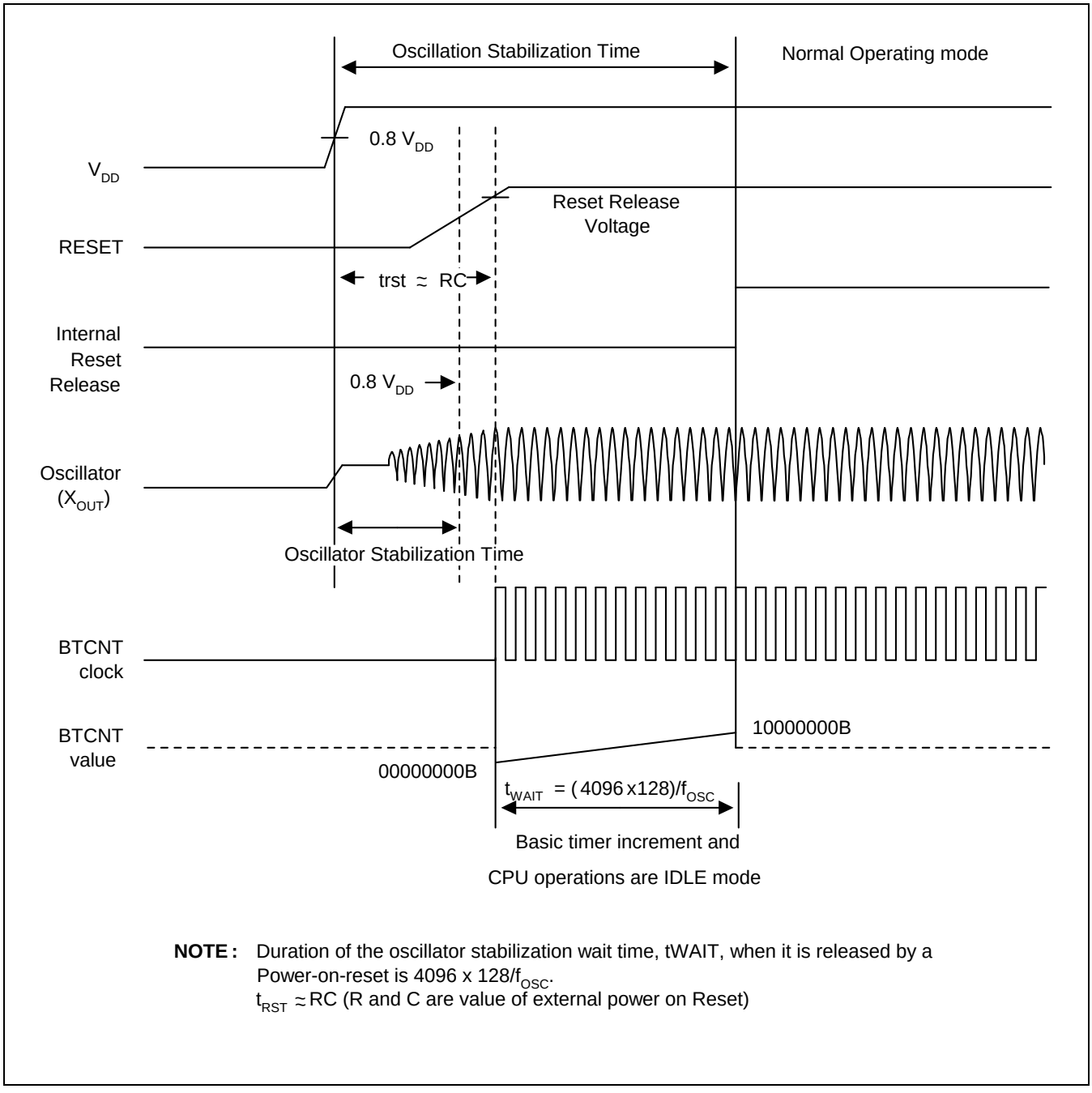
1. During Stop mode, an external power-on Reset or an external interrupt occurs to release the Stop mode and start oscillation.

2. If an external power-on Reset occurs, the basic timer counter will increase at the rate of fOSC/4096. If an external interrupt is used to release the Stop mode, the BTCNT value increases at the rate of preset clock source.

3. Clock oscillation stabilization interval begins and continues until the bit 7 of basic timer counter is set.

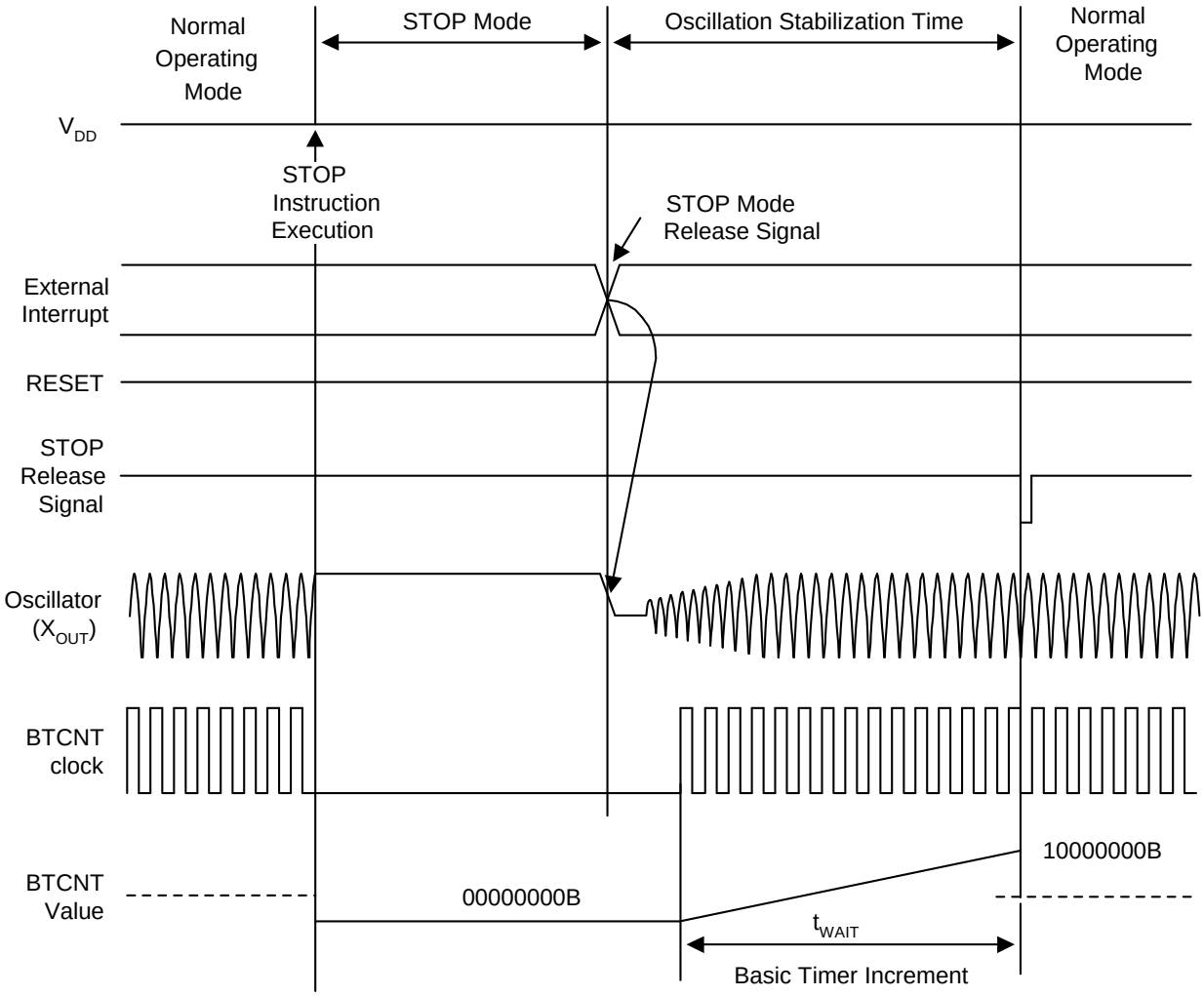
4. When a BTCNT.7 is set, normal CPU operation can resume.

Figure 10-2 and *Figure 10-3* show the oscillation stabilization time on RESET and STOP mode release.



NOTE : Duration of the oscillator stabilization wait time, tWAIT, when it is released by a Power-on-reset is 4096 x 128/f_{OSC}.
$t_{RST} \simeq$ RC (R and C are value of external power on Reset)

Figure 10-2 Oscillation Stabilization Time on RESET



NOTE: Duration of the oscillator stabilzation wait time, tWAIT,it is released by an interrupt is determined by the setting in basic timer control register, BTCON.

BTCON.3	BTCON.2	t_{WAIT}	t_{WAIT} (When f_{OSC} is 10 MHz)
0	0	(4096 x 128)/fosc	52.4 ms
0	1	(1024 x 128)/fosc	13.1 ms
1	0	(128 x 128)/fosc	1.63 ms
1	1	Invalid setting	—

Figure 10-3 Oscillation Stabilization Time on STOP Mode Release

Example 10-1 Programming Tip — Configuring the Basic Timer

This example shows how to configure the basic timer to sample specification.

```
            ORG    0000H
            VECTOR 00H, INT_94C4       ; S3F94C8/F94C4 has only one interrupt vector
;--------------<< Smart Option >>
            ORG    003CH
            DB     00H                 ; 003CH, must be initialized to 0
            DB     00H                 ; 003DH, must be initialized to 0
            DB     0E7H                ; 003EH, enable LVR (2.3 V)
            DB     03H                 ; 003FH, internal RC (3.2 MHz in VDD = 5 V)
;--------------<< Initialize System and Peripherals >>
            ORG    0100H
RESET:      DI                         ; Disable interrupt
            LD     CLKCON, #00011000B  ; Select non-divided CPU clock
            LD     SP, #0C0H           ; Stack pointer must be set
            •
            •
            LD     BTCON,#02H          ; Enable watchdog function
                                       ; Basic timer clock: fOSC/4096
                                       ; Clear Basic counter (BTCNT)
            •
            •
            •
            EI                         ; Enable interrupt
;--------------<< Main loop >>
MAIN:
            •
            LD     BTCON, #02H         ; Enable watchdog function
                                       ; Clear Basic counter (BTCNT)
            •
            •
            •
            JR     T, MAIN             ;
;--------------<< Interrupt Service Routines >>
INT_94C4:
            •                          ; Check Interrupt enable bit and Pending bit
            •                          ;
            •                          ; Clear Pending bit
            IRET                       ;
            •
            •
            END                        ;
```

10.3 Timer 0

10.3.1 Timer 0 Control Registers (T0CON)

The timer 0 control register, T0CON, selects the timer 0 operating mode (interval timer) and input clock frequency to clear the timer 0 counter and to enable the T0 match interrupt. It also contains a pending bit for T0 match interrupts.

A Reset clears T0CON to “00H”. This operation sets timer 0 to normal interval timer mode, selects an input clock frequency of fOSC /4096, and disables the T0 match interrupts. The T0 counter can be cleared at any time during normal operation by writing a “1” to T0CON.3.

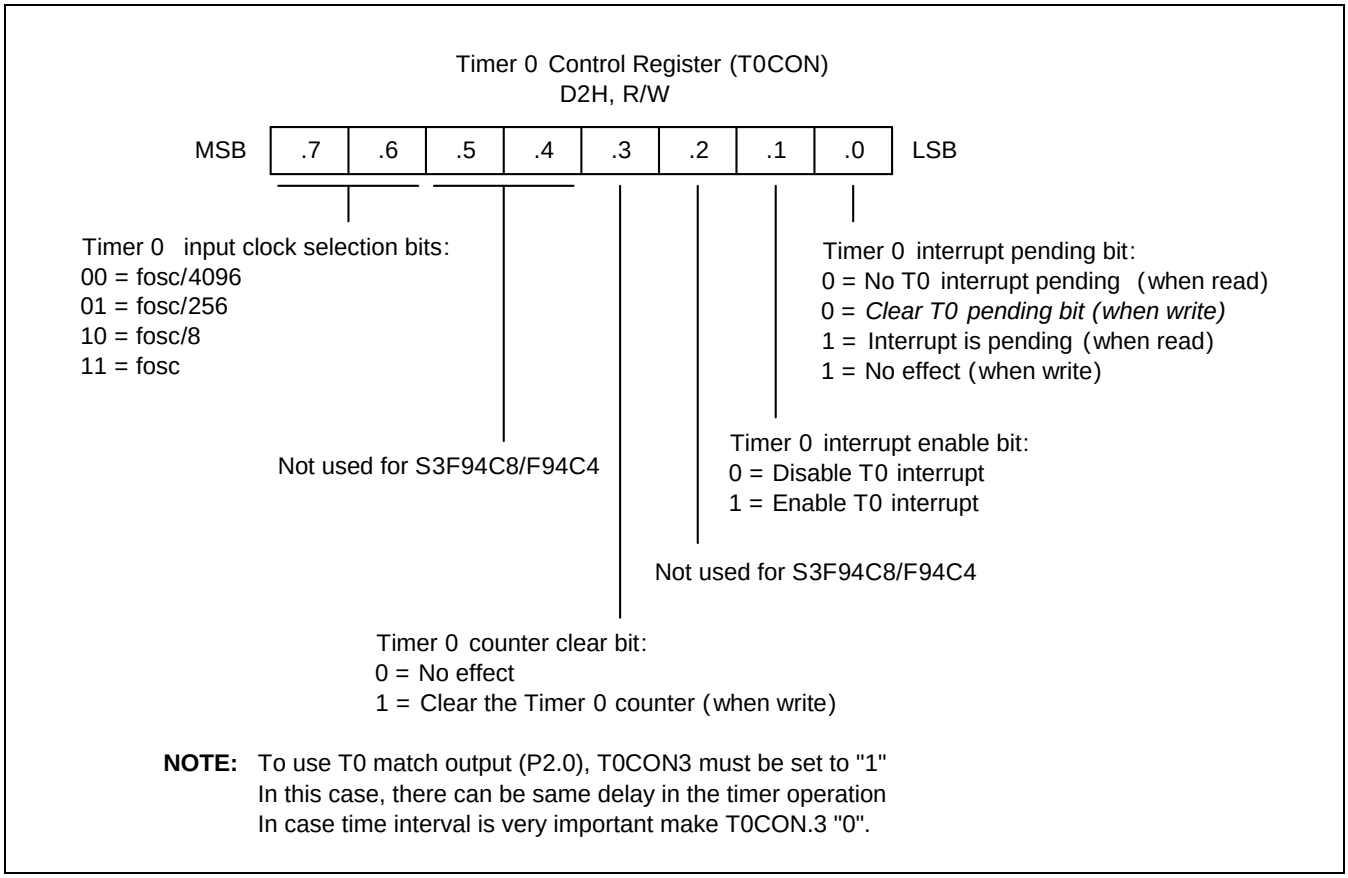
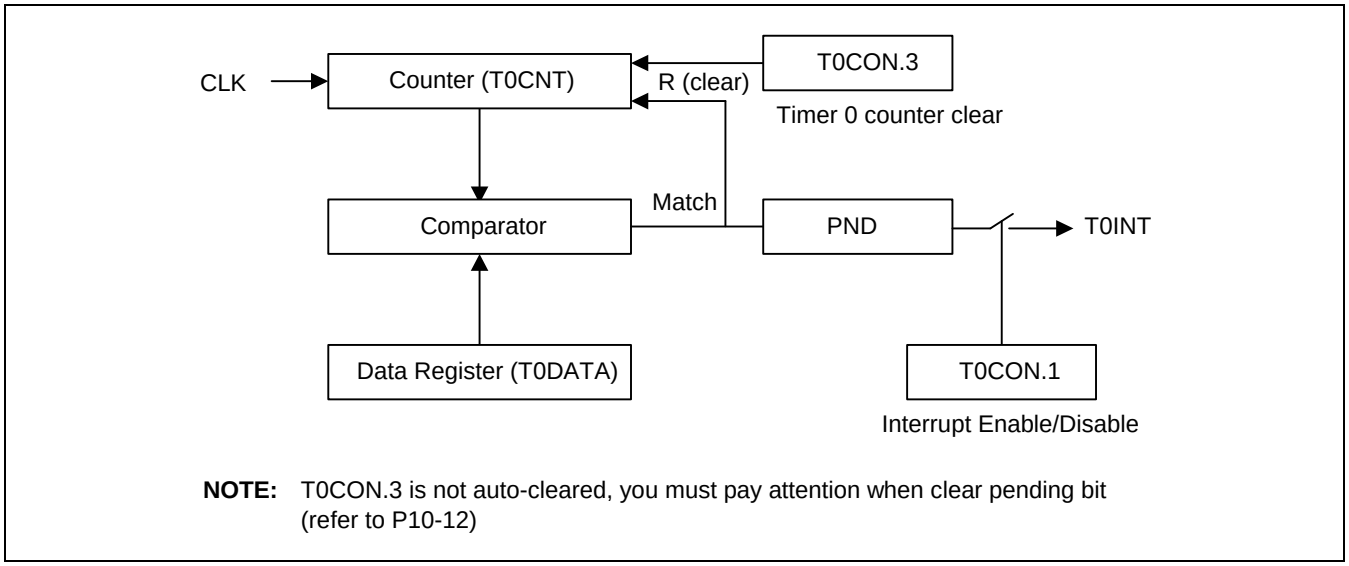

Timer 0 Control Register (T0CON)
D2H, R/W
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
Timer 0 input clock selection bits:
00 = fosc/4096
01 = fosc/256
10 = fosc/8
11 = fosc
Not used for S3F94C8/F94C4
Timer 0 interrupt pending bit:
0 = No T0 interrupt pending (when read)
0 = Clear T0 pending bit (when write)
1 = Interrupt is pending (when read)
1 = No effect (when write)
Timer 0 interrupt enable bit:
0 = Disable T0 interrupt
1 = Enable T0 interrupt
Not used for S3F94C8/F94C4
Timer 0 counter clear bit:
0 = No effect
1 = Clear the Timer 0 counter (when write)
NOTE: To use T0 match output (P2.0), T0CON3 must be set to "1"
In this case, there can be same delay in the timer operation
In case time interval is very important make T0CON.3 "0".

Figure 10-4 Timer 0 Control Registers (T0CON)

10.3.2 Timer 0 Function Description

10.3.2.1 Interval Timer Mode

In interval timer mode, a match signal is generated when the counter value is identical to the value written to Timer 0 reference data register, T0DATA. The match signal generates a Timer 0 match interrupt (T0INT, vector 00H) and then clears the counter. For example, if you write the value “10H” to T0DATA, the counter will increment until it reaches “10H”. At this point, the Timer 0 interrupt request is generated, the counter value is reset, and counting is resumed.



NOTE: T0CON.3 is not auto-cleared, you must pay attention when clear pending bit (refer to P10-12)

Figure 10-5 Simplified Timer 0 Function Diagram (Interval Timer Mode)

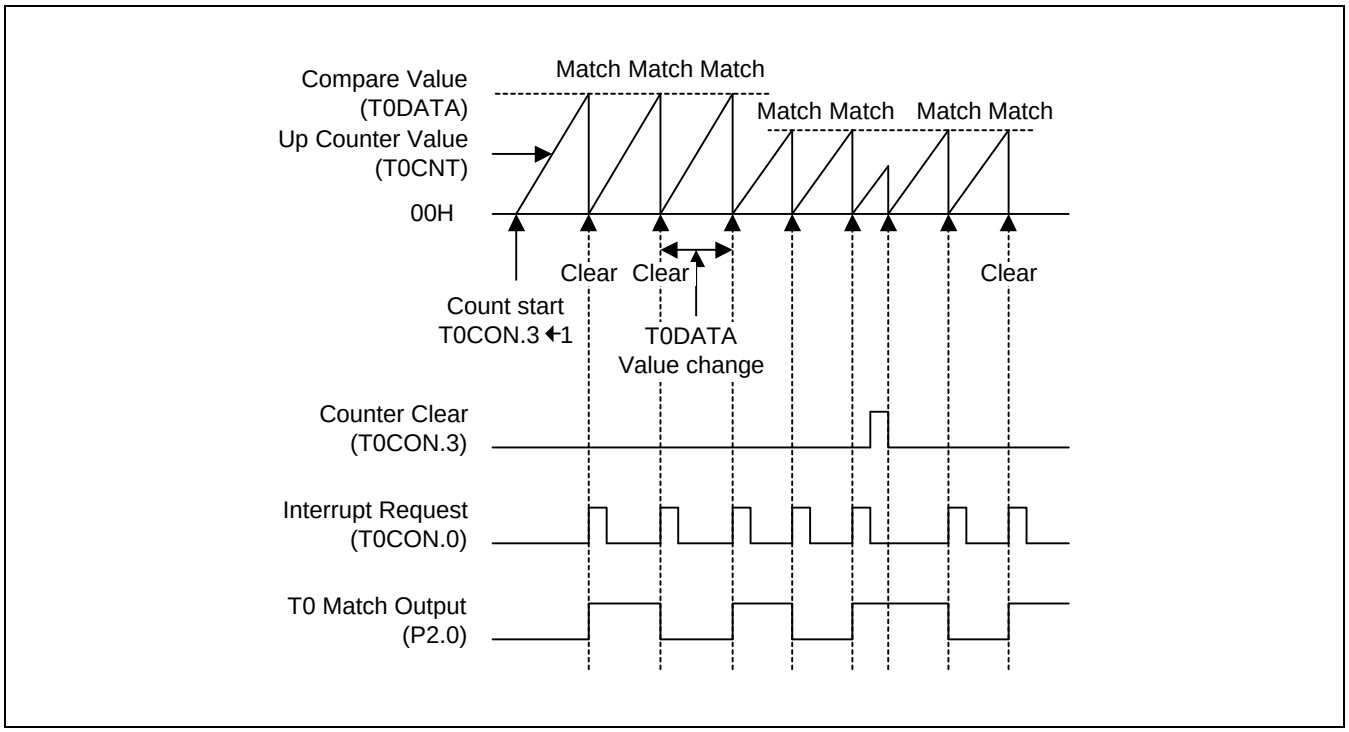
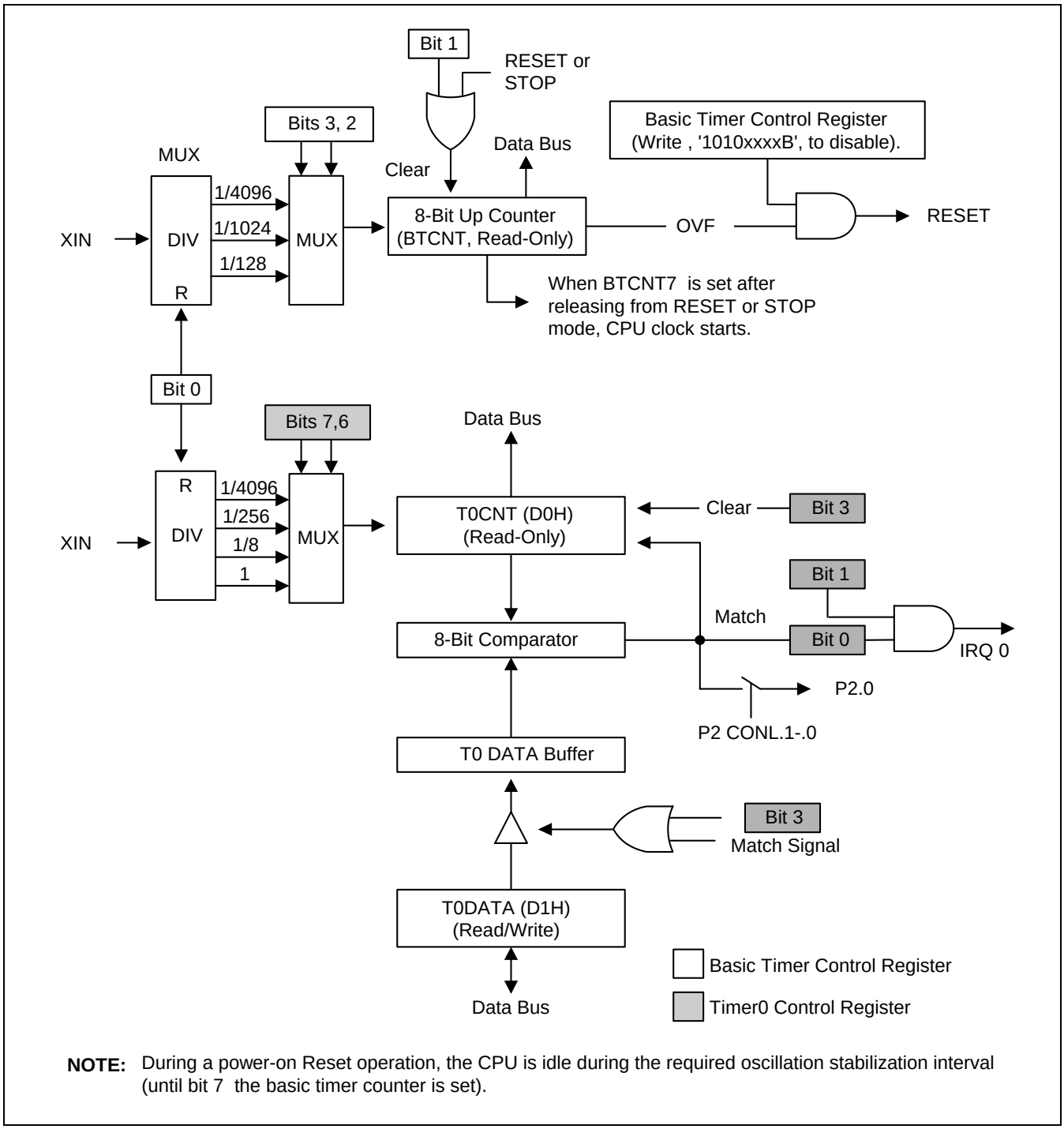


Figure 10-6 Timer 0 Timing Diagram



NOTE: During a power-on Reset operation, the CPU is idle during the required oscillation stabilization interval (until bit 7 the basic timer counter is set).

Figure 10-7 Basic Timer and Timer 0 Block Diagram

Example 10-2 Programming Tip — Configuring Timer 0 (Interval Mode)

The following sample program sets Timer 0 to interval timer mode.

```
            ORG    0000H
            VECTOR 00H, INT_94C4       ; S3F94C8/F94C4 has only one interrupt vector
            ORG    003CH
            DB     00H                 ; 003CH, must be initialized to 0
            DB     00H                 ; 003DH, must be initialized to 0
            DB     0E7H                ; 003EH, enable LVR (2.3 V)
            DB     03H                 ; 003FH, internal RC (3.2MHz in VDD = 5V)
            ORG    0100H
RESET:      DI                         ; Disable interrupt
            LD     BTCON,#10100011B    ; Disable watchdog
            LD     CLKCON,#00011000B   ; Select non-divided CPU clock
            LD     SP,#0C0H            ; Set stack pointer
            LD     P0CONH,#10101010B   ;
            LD     P0CONL,#10101010B   ; P0.0–0.7 push-pull output
            LD     P1CON,#00001010B    ; P1.0–P1.1 push-pull output
            LD     P2CONH,#01001010B   ;
            LD     P2CONL,#10101010B   ; P2.0–P2.6 push-pull output
;--------------<< Timer 0 settings >>
            LD     T0DATA, #50H        ; CPU=3.2MHz, interrupt interval=4 msec
            LD     T0CON, #01001010B   ; fOSC/256, Enable Timer 0 interrupt
            •
            •
            •
            EI                         ; Enable interrupt
;--------------<< Main loop >>
MAIN:       NOP                        ; Start main loop
            •
            •
            •
            CALL   LED_DISPLAY         ; Sub-block module
            •
            •
            •
            CALL   JOB                 ; Sub-block module
            •
            •
            •
            JR     T, MAIN             ;
LED_DISPLAY: NOP                       ;
            •                          ;
            •                          ;
            •                          ;
            RET                        ;
JOB:        NOP                        ;
            •                          ;
            •                          ;
            •                          ;
            RET                        ;
```

```
;--------------<< Interrupt Service Routines >>
INT_94C4:
            TM     T0CON,#00000010B    ; Check Interrupt enable
            JR     Z,NEXT_CHK1         ;
            TM     T0CON, #00000001B   ; If timer 0 interrupt occurs,
            JP     NZ,INT_TIMER0       ; T0CON.0 bit will be set.
NEXT_CHK1:
            •                          ; Interrupt enable bit and pending bit check
            •                          ;
            •                          ;
            IRET                       ;
INT_TIMER0:                            ; Timer 0 interrupt service routine
            •
            •
            •
            AND    T0CON, #11110110B   ; Clear Pending bit
            IRET                       ;
            •
            •
            END                        ;
```

11 PULSE WIDTH MODULATION

11.1 Overview of Pulse Width Modulation

The S3F94C8/F94C4 microcontroller consists of the Pulse Width Modulation (PWM) circuit. This circuit can be configured in one of the three resolutions:

- 8-bit resolution: 6-bit base + 2-bit extension
- 12-bit resolution: 6-bit base + 6-bit extension
- 14-bit resolution: 8-bit base + 6-bit extension

These three resolutions are mutually exclusive, that is, only one resolution can work at any time. PWMEX.1-.0 selects the resolution to be used.

The operation of all PWM circuits is controlled by a single control register, PWMCON.

The PWM counter is an incrementing counter used by the PWM circuits. To start the counter and enable the PWM circuits, set PWMCON.2 to “1”. If the counter is stopped, it retains the current count value. When the counter is re-started, it resumes counting from the retained count value. To clear the counter, set PWMCON.3 to “1”.

You can select a clock for the PWM counter by setting PWMCON.6-.7. The clocks that you can select are fOSC /64, fOSC /8, fOSC /2, and fOSC /1.

11.1.1 Function Description

11.1.1.1 PWM

The PWM circuits have the following components:

- PWM mode selection (PWMEX.1-.0)
- Base comparator and extension cycle circuit
- Base reference data registers (PWMDATA, PWMDATA1)
- Extension data registers (PWMEX)
- PWM output pins (P0.6/PWM)

11.1.1.2 PWM Counter

The PWM counter is an incrementing counter comprised of a lower base counter and an upper extension counter.

To determine the PWM module’s base operating frequency, the lower base counter is compared to the PWM base data register value. In order to achieve higher resolutions, the extension bits of upper counter can be used to modulate the “stretch” cycle. To control the “stretching” of PWM output duty cycle at specific intervals, the extended counter value is compared to the value you write to the module's extension bits.

11.1.1.3 PWM Data and Extension Registers

PWM (duty) data consist of base data bits and extension data bits. Determine the output value generated by the PWM circuit. For each PWM resolution, the location of base data bits and extension data bits are different combinations of register PWMDATA (F2H), PWMDATA1 (F0H), and PWMEX (F1H):

- 8-bit resolution, 6-bit base + 2-bit extension:
  - Base 6 data bits: PWMDATA.7-.2
  - Extension 2 bits: PWMDATA.1-.0
- 12-bit resolution, 6-bit base + 6-bit extension:
  - Base 6 data bits: PWMDATA1.5-.0
  - Extension 6 bits: PWMEX.7-.2
- 14-bit resolution, 8-bit base + 6-bit extension:
  - Base 8 data bits: PWMDATA1.7-.0.
  - Extension 6 bits: PWMEX.7-.2

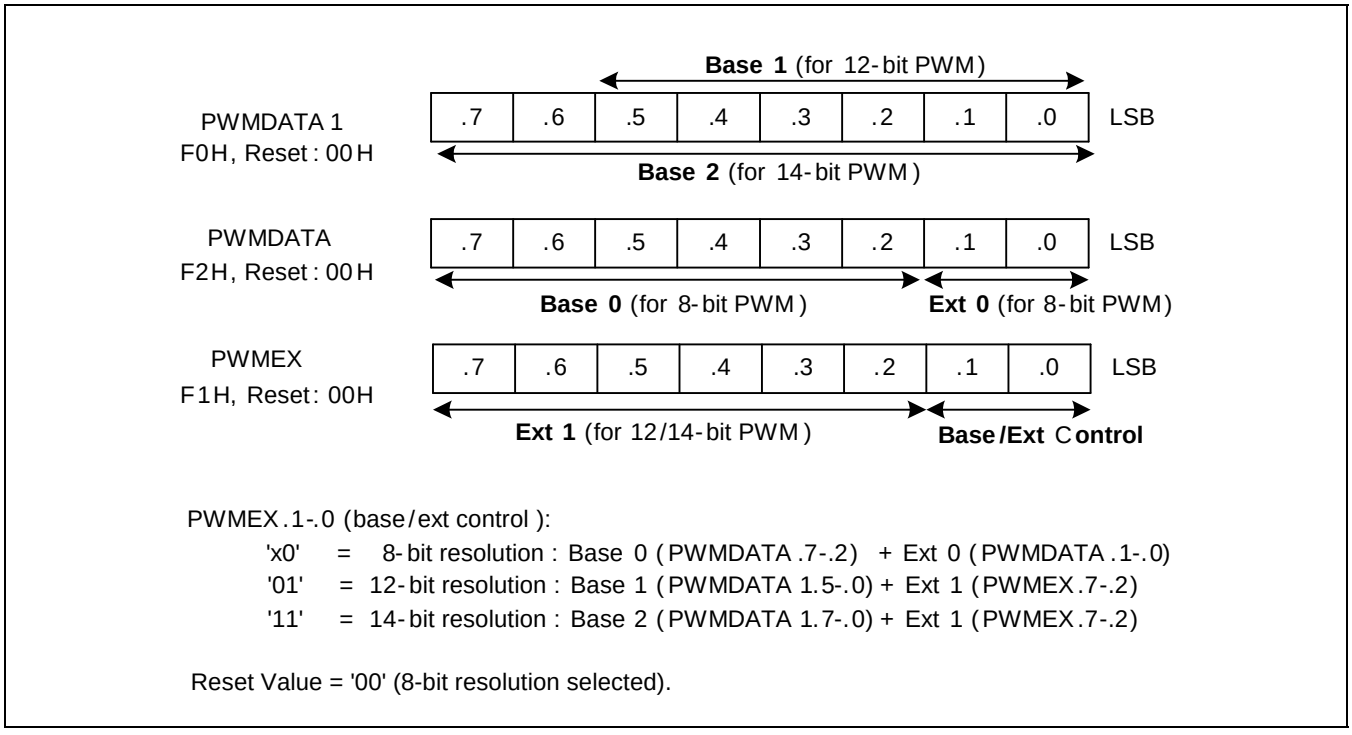

Base 1 (for 12-bit PWM)
PWMDATA 1
F0H, Reset : 00H
.7 .6 .5 .4 .3 .2 .1 .0
LSB
Base 2 (for 14-bit PWM)
PWMDATA
F2H, Reset : 00H
.7 .6 .5 .4 .3 .2 .1 .0
LSB
Base 0 (for 8-bit PWM)
Ext 0 (for 8-bit PWM)
PWMEX
F1H, Reset : 00H
.7 .6 .5 .4 .3 .2 .1 .0
LSB
Ext 1 (for 12/14-bit PWM)
Base/Ext Control
PWMEX.1-.0 (base/ext control):
'x0' = 8-bit resolution : Base 0 (PWMDATA.7-.2) + Ext 0 (PWMDATA.1-.0)
'01' = 12-bit resolution : Base 1 (PWMDATA1.5-.0) + Ext 1 (PWMEX.7-.2)
'11' = 14-bit resolution : Base 2 (PWMDATA1.7-.0) + Ext 1 (PWMEX.7-.2)
Reset Value = '00' (8-bit resolution selected).

Figure 11-1 PWM Data and Extension Registers

To program the required PWM output, load the appropriate initialization values into the data registers (PWMDATA) and the extension registers (PWMEX). To start the PWM counter or to resume the counting, set PWMCON.2 to “1”.

A reset operation disables all PWM output. The current counter value is retained when the counter stops. When the counter starts, counting resumes at the retained value.

11.1.1.4 PWM Clock Rate

The timing characteristic of PWM output is based on the fOSC clock frequency. The PWM counter clock value is determined by setting PWMCON.6–.7.

Table 11-1 PWM Control and Data Registers

Register Name	Mnemonic	Address	Function
PWM data registers	PWMDATA	F2H	PWM waveform output setting registers
	PWMDATA1	F0H	
	PWMEX	F1H	
PWM control registers	PWMCON	F3H	PWM counter stop/start (resume) and fOSC clock setting

11.1.1.5 PWM Function Description

The PWM output signal toggles to Low level whenever the lower base counter matches the reference value stored in the module's data register (PWMDATA). If the value in the PWMDATA register is not zero, an overflow of the lower counter causes the PWM output to toggle to High level. In this way, the reference value written to the data register determines the module's base duty cycle.

The value in the extension counter is compared to the extension settings in the extension data bits. This extension counter value, together with extension logic and the PWM module's extension bits, is then used to "stretch" the duty cycle of the PWM output. The "stretch" value is one extra clock period at specific intervals or cycles (see *Table 11-2* for more information).

For example, if the value in the extension register is '04H' in 8-bit base + 6-bit extension mode, the 32nd cycle will be one pulse longer than the other 63 cycles. If the base duty cycle is 50%, the duty of the 32nd cycle will therefore be "stretched" to approximately 51% duty.

If you write 80H to the extension register, all odd-numbered cycles will be one pulse longer.

If you write FCH to the extension register, all cycles will be stretched by one pulse (except the 64th cycle).

PWM output goes to an output buffer and then to the corresponding PWM output pin. In this way, you can obtain high output resolution at high frequencies.

11.1.1.6 PWM Output Waveform

- 6-bit base + 2-bit extension mode:

Table 11-2 PWM output "stretch" Values for Extension Data bits Ext0 (PWMDATA.1–.0)

PWMDATA Bit (Bit1–Bit0)	"Stretched" Cycle Number
00	–
01	2
10	1, 3
11	1, 2, 3

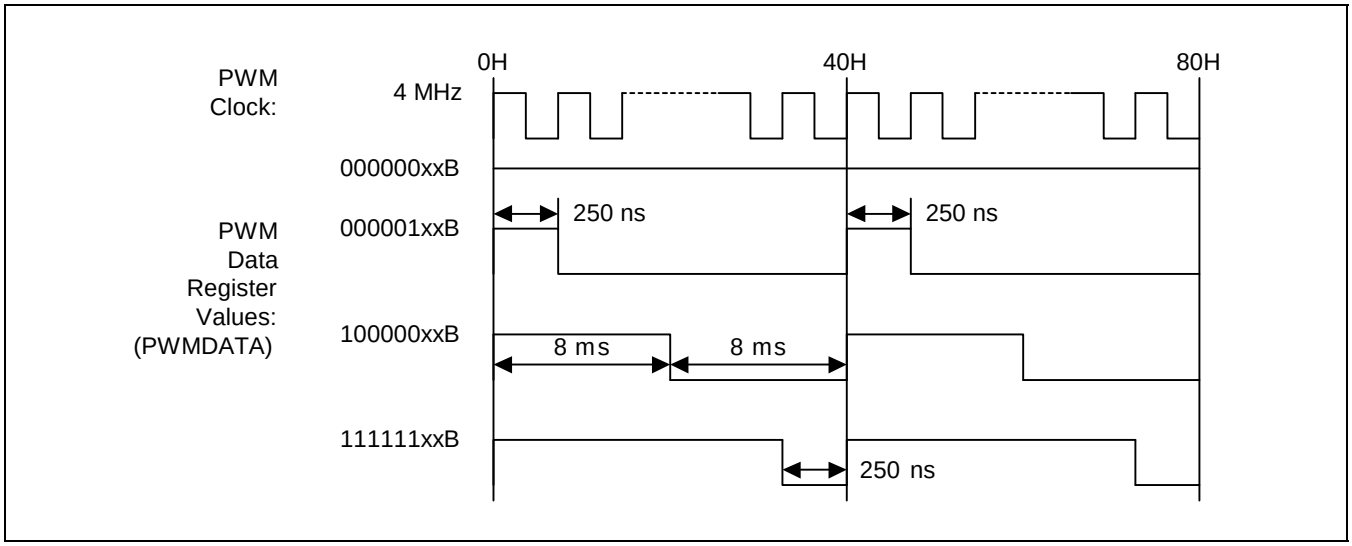


Figure 11-2 PWM Basic Waveform (6-bit base)

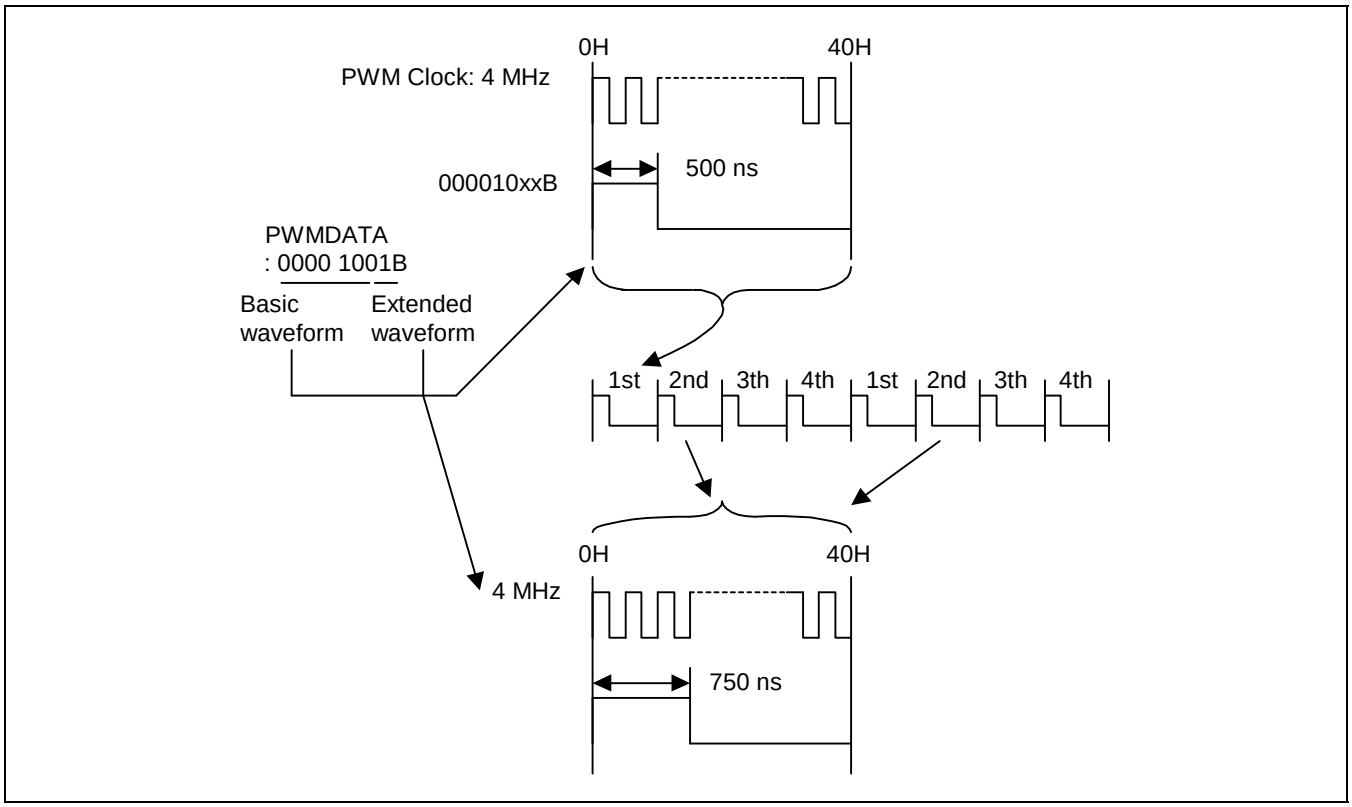


Figure 11-3 Extended PWM Waveform (6-bit base + 2-bit extension)

- 6-bit base + 6-bit extension mode:

Table 11-3 PWM output "stretch" Values for Extension Data bits Ext1 (PWMEX.7-.2)

PWMEX Bit	"Stretched" Cycle Number
7	1, 3, 5, 7, 9, . . . , 55, 57, 59, 61, 63
6	2, 6, 10, 14, . . . , 50, 54, 58, 62
5	4, 12, 20, . . . , 44, 52, 60
4	8, 24, 40, 56
3	16, 48
2	32

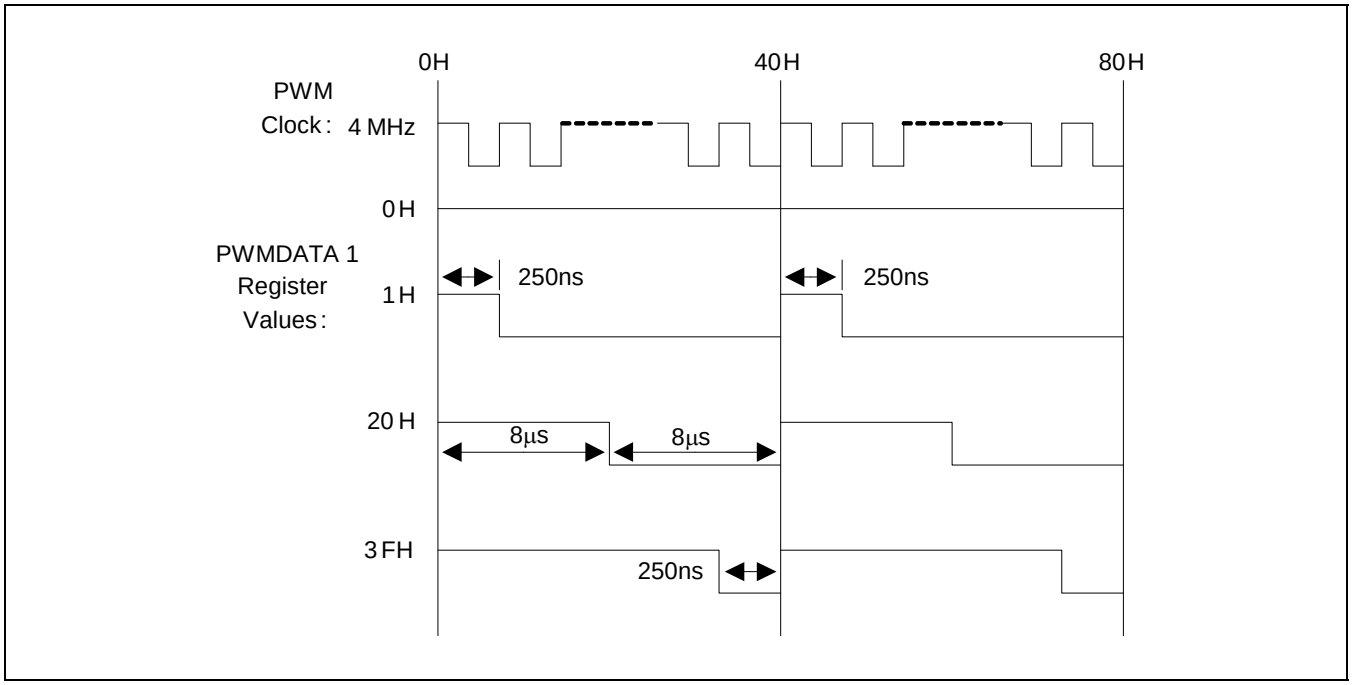


Figure 11-4 PWM Basic Waveform (6-bit base)

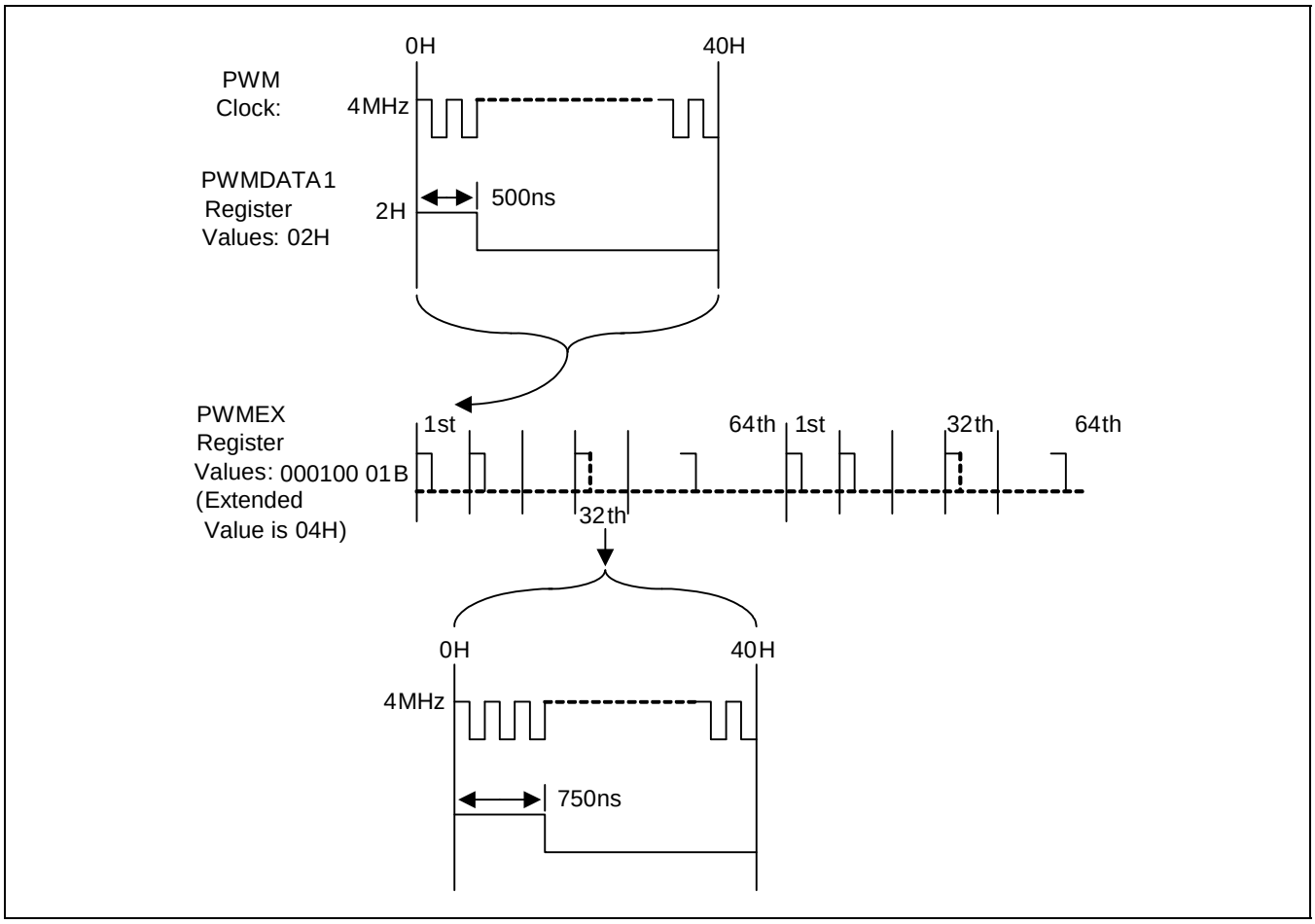


Figure 11-5 Extended PWM Waveform (6-bit base + 6-bit extension)

- 8-bit base + 6-bit extension mode:

Table 11-4 PWM output "stretch" Values for Extension Data bits Ext1 (PWMEX.7-.2)

PWMEX Bit	"Stretched" Cycle Number
7	1, 3, 5, 7, 9, . . . , 55, 57, 59, 61, 63
6	2, 6, 10, 14, . . . , 50, 54, 58, 62
5	4, 12, 20, . . . , 44, 52, 60
4	8, 24, 40, 56
3	16, 48
2	32

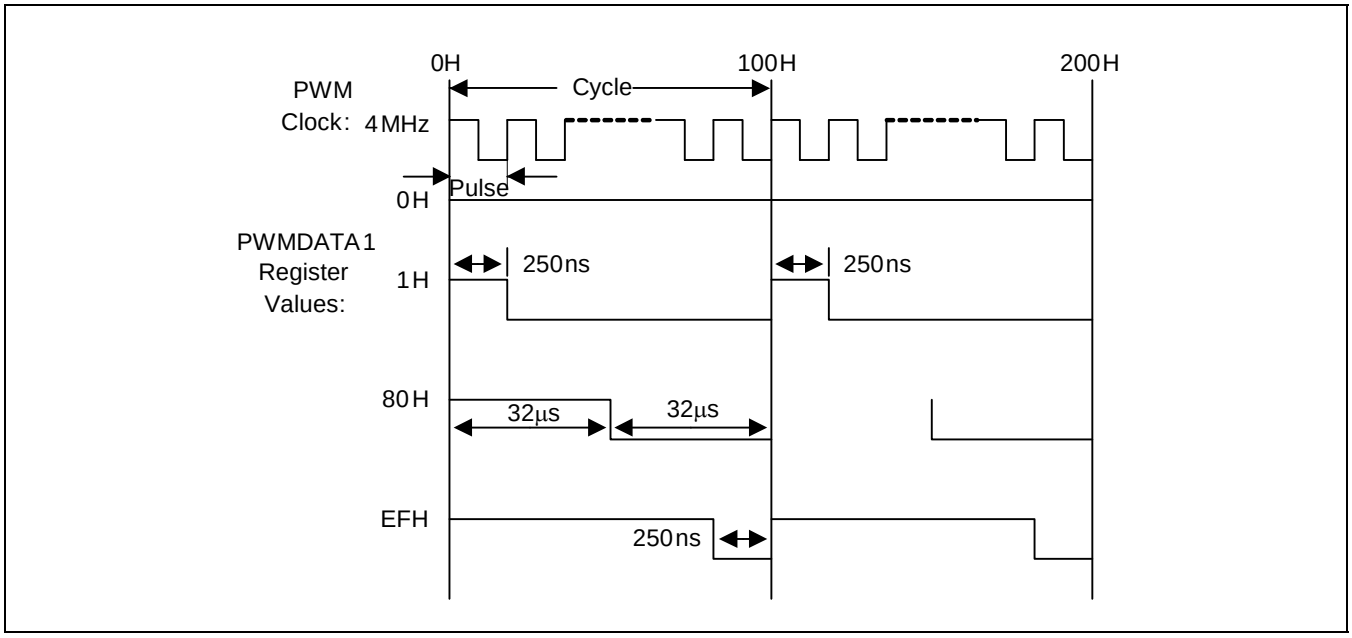


Figure 11-6 PWM Basic Waveform (8-bit base)

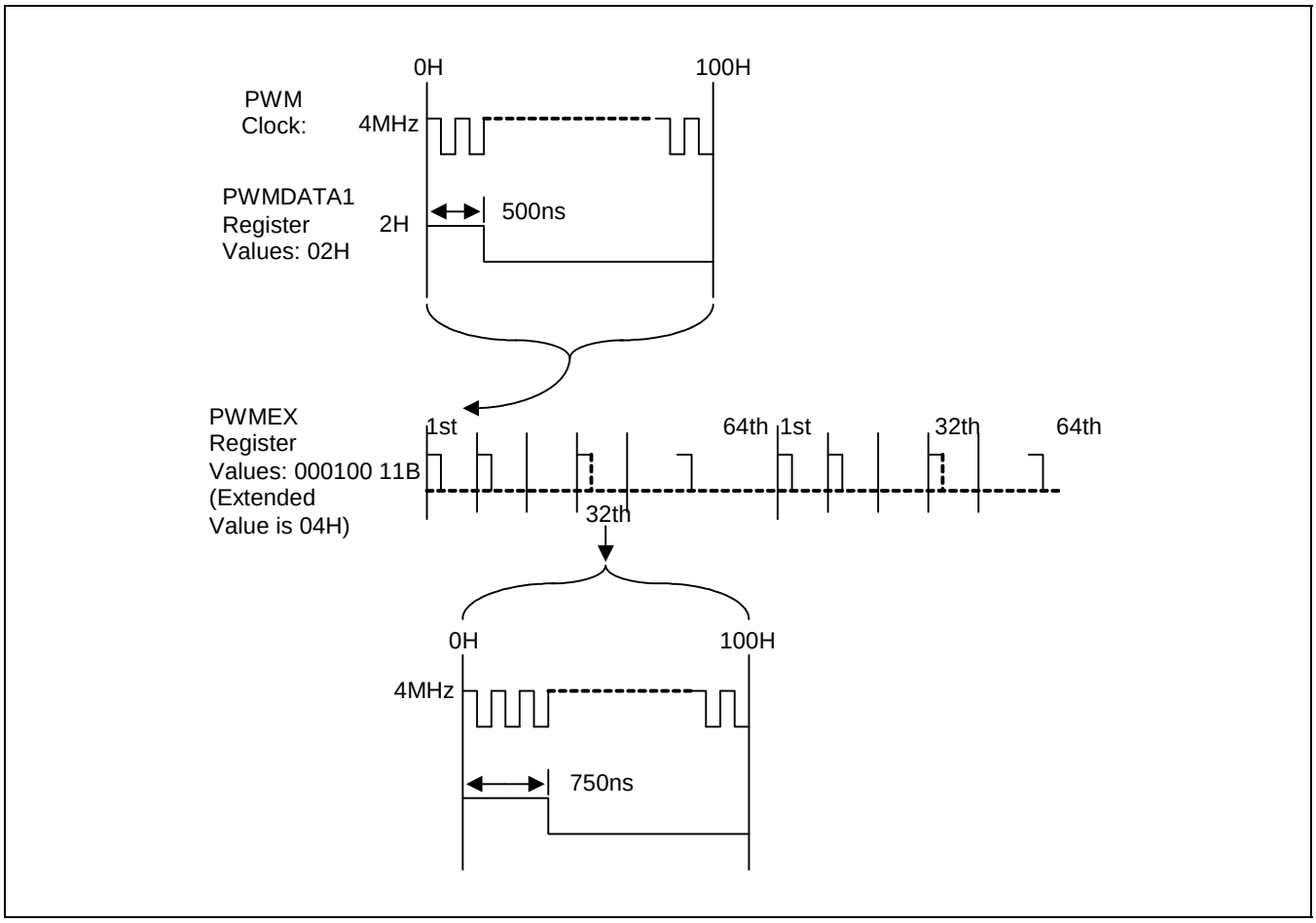


Figure 11-7 PWM Basic Waveform (8-bit base + 6-bit extension)

11.1.2 PWM Control Register (PWMCON)

The control register for the PWM module, PWMCON, is located at register address F3H. PWMCON is used for all three PWM resolutions. Bit settings in the PWMCON register control the following functions:

- PWM counter clock selection
- PWM data reload interval selection
- PWM counter clear
- PWM counter stop/start (or resume) operation
- PWM counter overflow (upper counter overflow) interrupt control

A reset clears all PWMCON bits to logic zero and disables the entire PWM module.

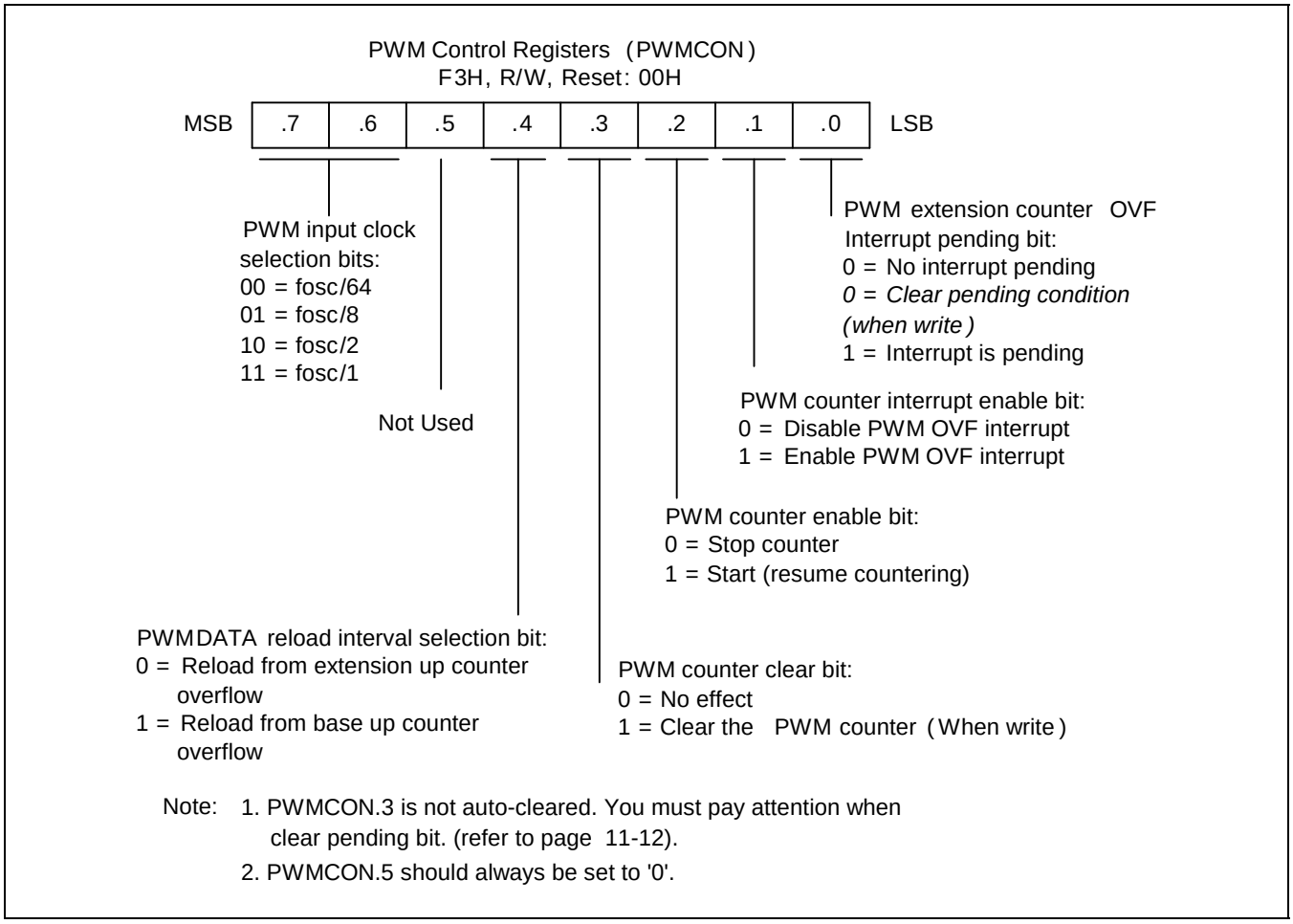

PWM Control Registers (PWMCON)
F3H, R/W, Reset: 00H
MSB
.7
.6
.5
.4
.3
.2
.1
.0
LSB
PWM input clock selection bits:
00 = fosc/64
01 = fosc/8
10 = fosc/2
11 = fosc/1
Not Used
PWMDATA reload interval selection bit:
0 = Reload from extension up counter overflow
1 = Reload from base up counter overflow
PWM counter clear bit:
0 = No effect
1 = Clear the PWM counter (When write)
PWM counter enable bit:
0 = Stop counter
1 = Start (resume countering)
PWM counter interrupt enable bit:
0 = Disable PWM OVF interrupt
1 = Enable PWM OVF interrupt
PWM extension counter OVF Interrupt pending bit:
0 = No interrupt pending
0 = Clear pending condition (when write)
1 = Interrupt is pending
Note: 1. PWMCON.3 is not auto-cleared. You must pay attention when clear pending bit. (refer to page 11-12).
2. PWMCON.5 should always be set to '0'.

Figure 11-8 PWM Control Register (PWMCON)

11.1.3 PWM Extension Register (PWMEX)

The extension register for the PWM module, PWMEX, is located at register address F1H. PWMEX is used for the purpose of resolution selection and extension bits of 6+6 and 8+6 resolution. Bit settings in the PWMEX register control the following functions:

- PWM extension bits for 6+6 resolution and 8+6 resolution mode
- PWM resolution selection

A reset clears all PWMEX bits to logic zero, and choose 6+2 resolution without extension.

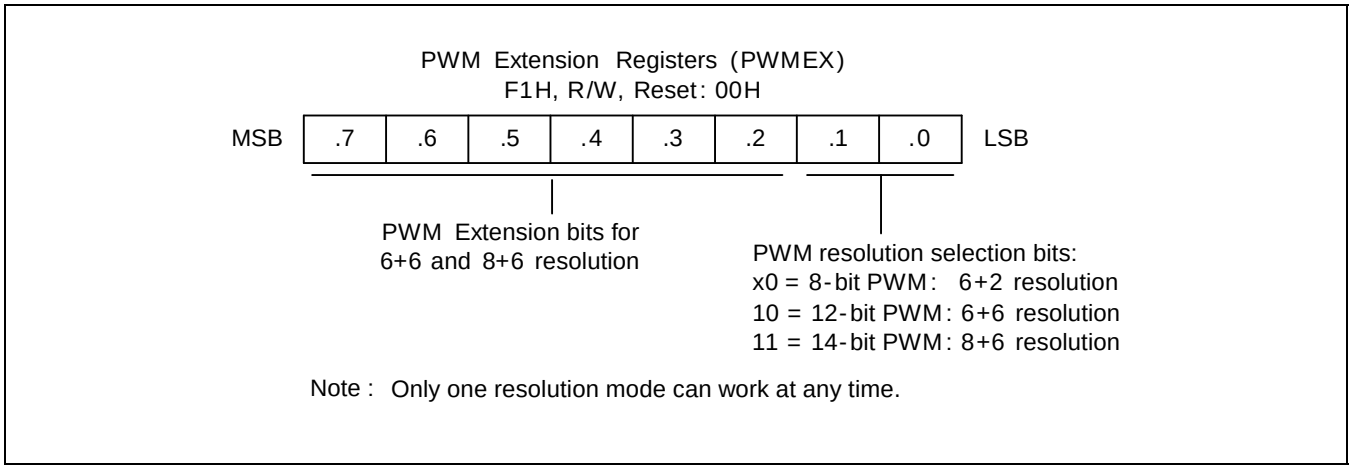


Figure 11-9 PWM Extension Register (PWMEX)

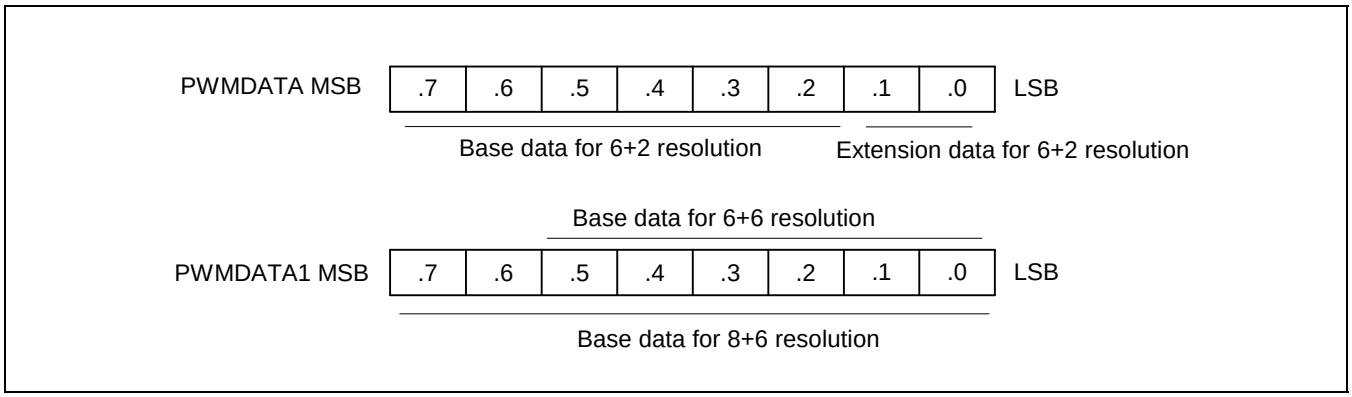


Figure 11-10 PWM Data Register (PWMDATA)

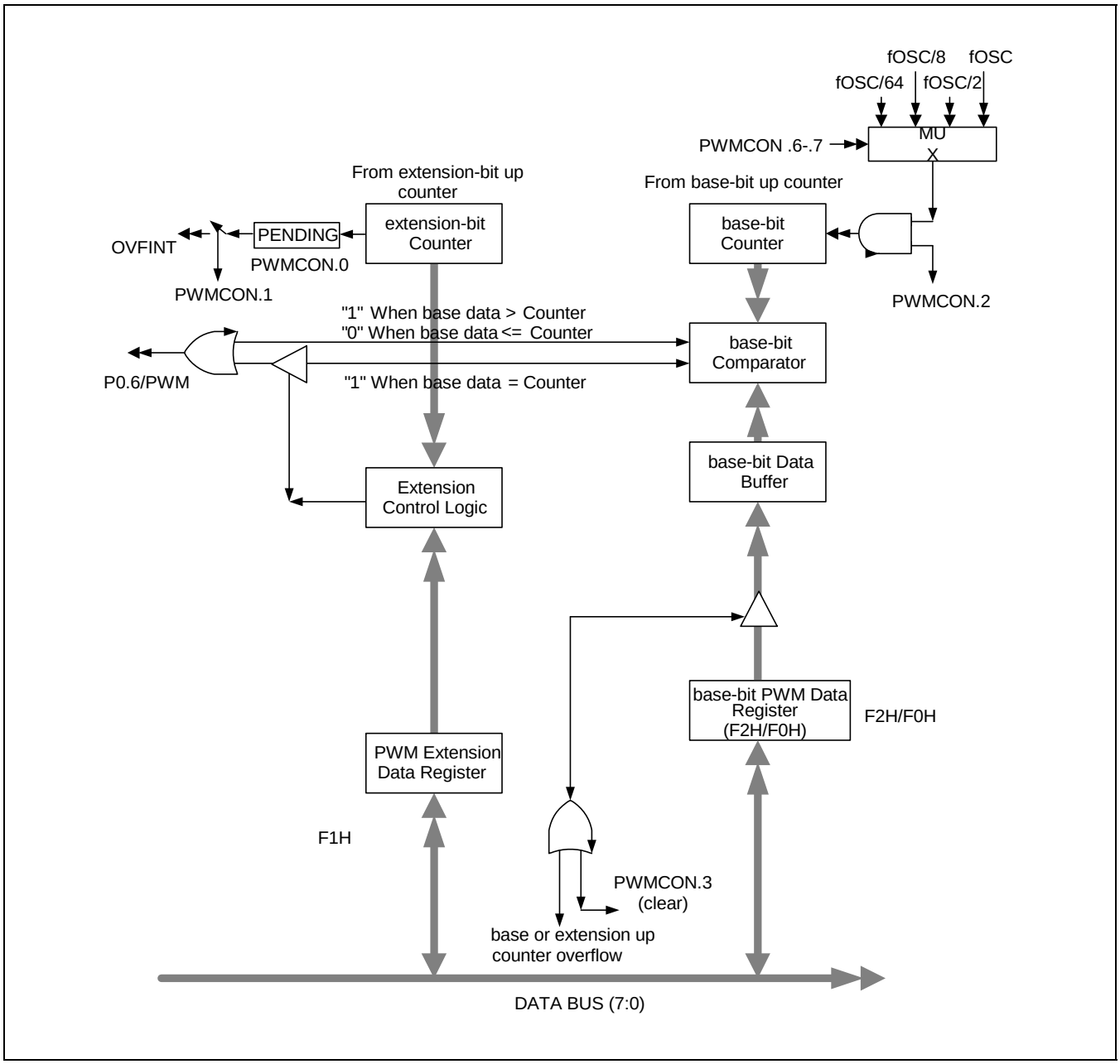


Figure 11-11 PWM Module Functional Block Diagram

Example 11-1 Programming Tip — Programming the PWM Module to Sample Specifications

```
;--------------<< Interrupt Vector Address >>
            VECTOR 00H, INT_94C4       ; S3F94C8/F94C4 interrupt vector
;--------------<< Smart Option >>
            ORG    003CH
            DB     000H                ; 003CH, must be initialized to 1.
            DB     000H                ; 003DH, must be initialized to 1.
            DB     0FFH                ; 003EH, Enable LVR (2.3)
            DB     000H                ; 003FH, External Crystal Oscillator
;--------------<< Initialize System and Peripherals >>
            ORG    0100H
RESET:      DI                         ; Disable Interrupt
            LD     BTCON,#10100011B    ; Disable Watchdog
            •
            •

            LD     PWMEX,#00000000B    ; Configure PWM as 6-bit base + 2-bit extension
            LD     P0CONH,#10011010B   ; Configure P0.6 PWM output
            LD     PWMCON,#00000110B   ; fOSC/64, Enable counter/interrupt
            AND    PWMEX,#00000011B    ; Set extension bits as 00 (basic output)
            LD     PWMDATA,#80H        ;
            •
            •
            EI                         ; Enable interrupt
;--------------<< Main loop >>
MAIN:                                  ;
            •                          ;
            •                          ;
            •                          ;
            •                          ;
            JR     t,MAIN              ;
INT_94C4:                              ; 94C4 interrupt service routine
            •
            •
            •
            AND    PWMCON,#11110110B   ; Clear pending bit
            IRET                       ;
            •
            •
            END
```

12 A/D CONVERTER

12.1 Overview of A/D Converter

The 10-bit A/D converter (ADC) module uses successive approximation logic to convert analog levels entering at one of the nine input channels to equivalent 10-bit digital values. The analog input level must lie between the VDD and VSS values.

The A/D converter has the following components:

- Analog comparator with successive approximation logic
- D/A converter logic
- ADC control register (ADCON)
- Nine multiplexed analog data input pins (ADC0–ADC8)
- 10-bit A/D conversion data output register (ADDATAH/L)

To initiate an analog-to-digital conversion procedure, write the channel selection data in the A/D converter control register ADCON. This selects one of the nine analog input pins (ADCn, n = 0–8) and sets the conversion start or enable bit, ADCON.0. The read-write ADCON register is located at address F7H.

During a normal conversion, ADC logic sets the successive approximation register to 200H (approximate half-way point of a 10-bit register). This register is then updated automatically during each conversion step. The successive approximation block performs 10-bit conversions for one input channel at a time. You can dynamically select different channels by manipulating the channel selection bit value (ADCON.7–4) in the ADCON register.

To start A/D conversion, set the enable bit, ADCON.0. When a conversion is completed, the end-of-conversion (EOC) bit, ACON.3, is automatically set to 1 and the result is dumped into the ADDATA register where it can be read. The A/D converter then enters into an idle state. Remember to read the contents of ADDATA before another conversion starts. Otherwise, the previous result will be overwritten by next conversion result.

NOTE: Since the ADC does not use sample-and-hold circuitry, it is important that any fluctuations in the analog level at ADC0–ADC8 input pins during a conversion procedure be kept to an absolute minimum. Any change in the input level, perhaps due to circuit noise, will invalidate the result.

12.1.1 Using A/D Pins for Standard Digital Input

The input pins in ADC module are alternatively used as digital input in port 0 and P2.6.

12.1.2 A/D Converter Control Register (ADCON)

The A/D converter control register, ADCON, is located at address F7H. It has four functions:

- Bits 7-4 select an analog input pin (ADC0–ADC8).
- Bit 3 indicates the status of A/D conversion.
- Bits 2-1 select the conversion speed.
- Bit 0 starts the A/D conversion.

Only one analog input channel can be selected at a time. You can dynamically select any one of the nine analog input pins (ADC0–ADC8) by manipulating the 4-bit value for ADCON.7–ADCON.4.

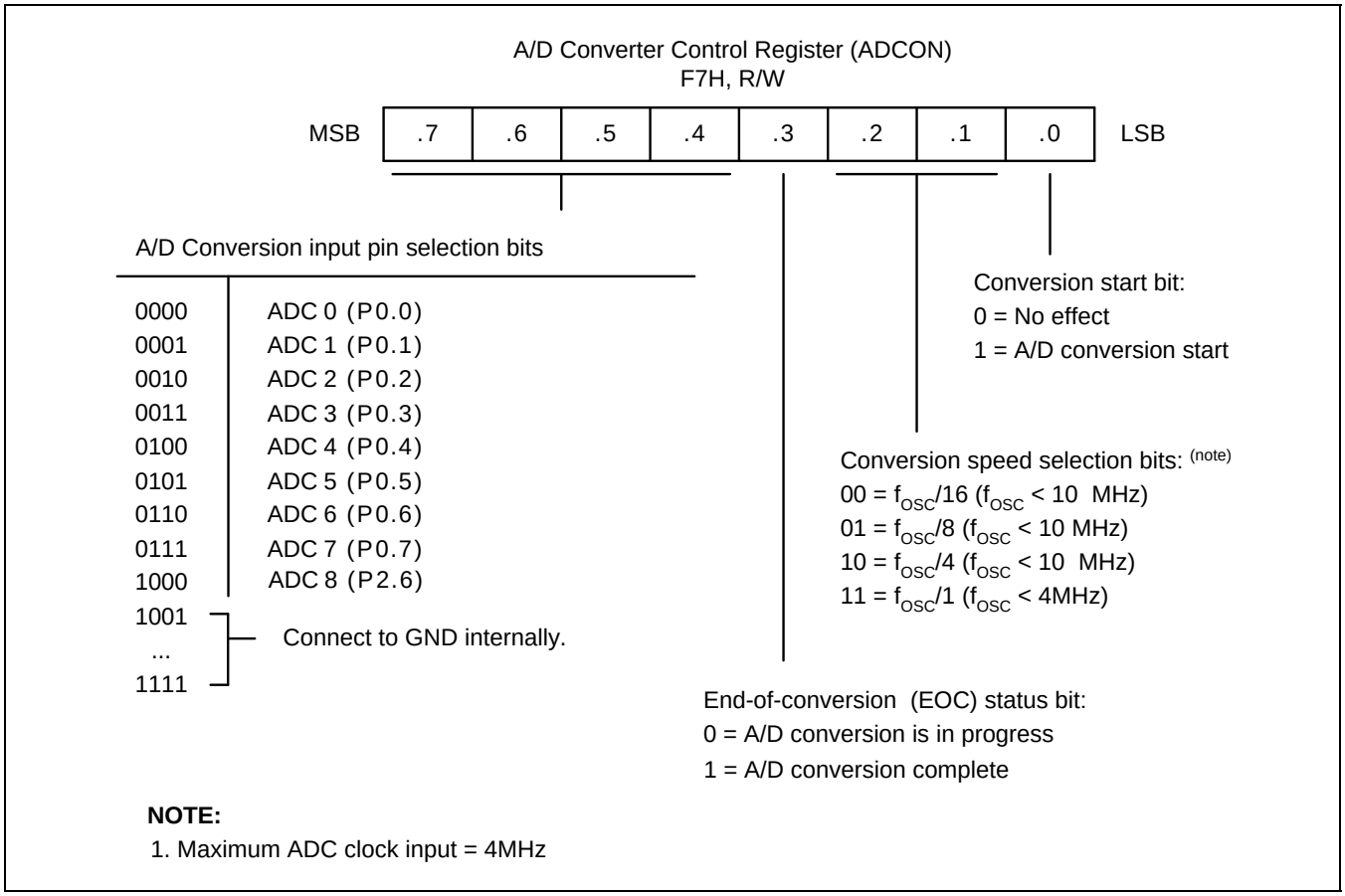


Figure 12-1 A/D Converter Control Register (ADCON)

12.1.3 Internal Reference Voltage Levels

In the ADC function block, the analog input voltage level is compared to the reference voltage. The analog input level must remain within the range of VSS to VDD.

Different reference voltage levels are generated internally along the resistor tree during analog conversion process for each conversion step. The reference voltage level for the first bit conversion is always 1/2 VDD.

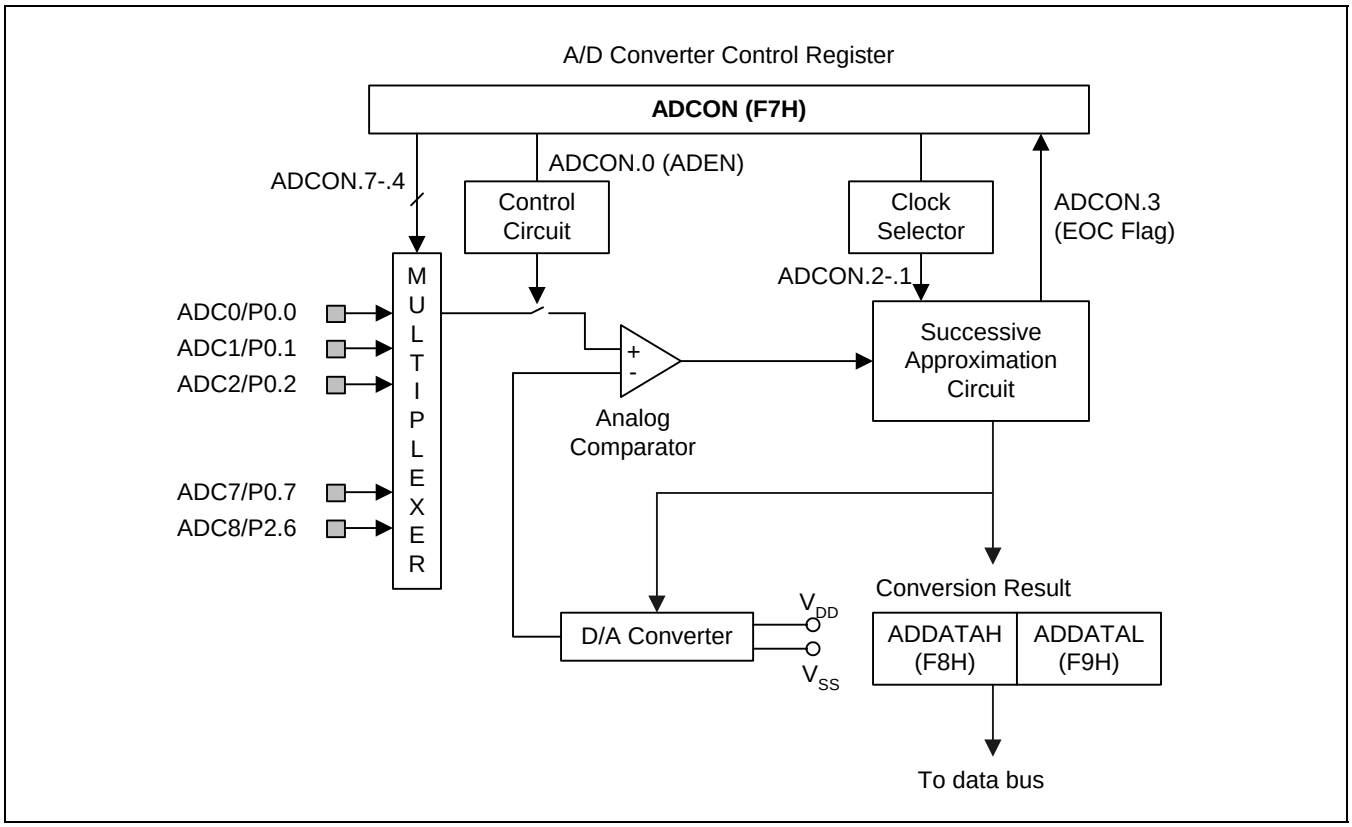


Figure 12-2 A/D Converter Circuit Diagram

Figure 12-3 A/D Converter Data Register (ADDATAH/L)

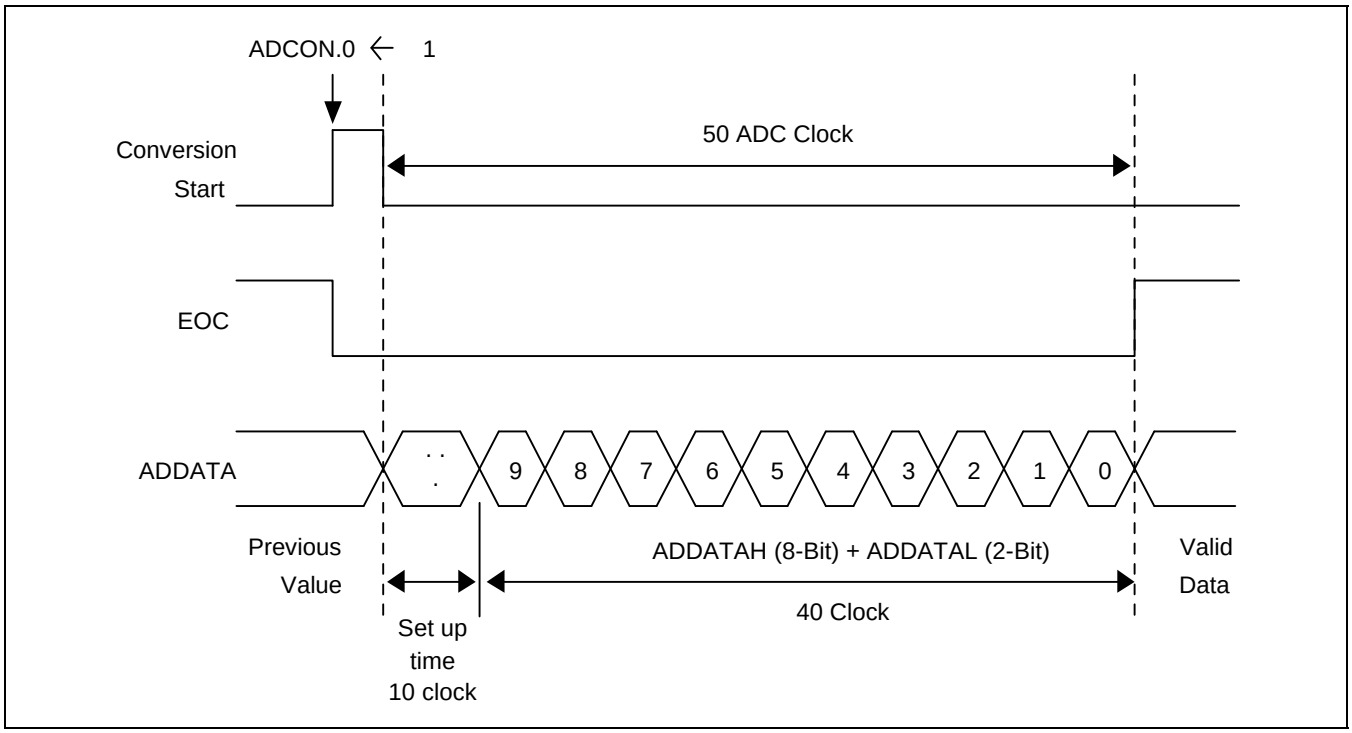


Figure 12-4 A/D Converter Timing Diagram

12.1.4 Conversion Timing

The A/D conversion process requires four steps (4 clock edges) to convert each bit and 10 clocks to step-up A/D conversion. Therefore, 50 clocks are required to complete a 10-bit conversion.

With a 10 MHz CPU clock frequency, one clock cycle is 400 ns (4/fxx). If each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

4 clocks/bit x 10-bit + step-up time (10 clock) = 50 clocks
50 clock x 400 ns = 20 μs at 10 MHz, 1 clock time = 4/fxx (assuming ADCON.2–.1 = 10)

12.1.5 Internal A/D Conversion Procedure

1. Analog input must remain between the voltage range of VSS and VDD.
2. Configure the analog input pins for input mode by making the appropriate settings in P0CONH, P0CONL, and P2CONH registers.
3. Before the conversion operation starts, you must first select one of the nine input pins (ADC0–ADC8) by writing the appropriate value to the ADCON register.
4. When conversion has been completed, (50 clocks have elapsed), the EOC flag is set to “1”, so that a check can be made to verify that the conversion was successful.
5. The converted digital value is loaded to the output register, ADDATAH (8-bit) and ADDATAL (2-bit), and then the ADC module enters into an idle state.
6. The digital conversion result can now be read from the ADDATAH and ADDATAL register.

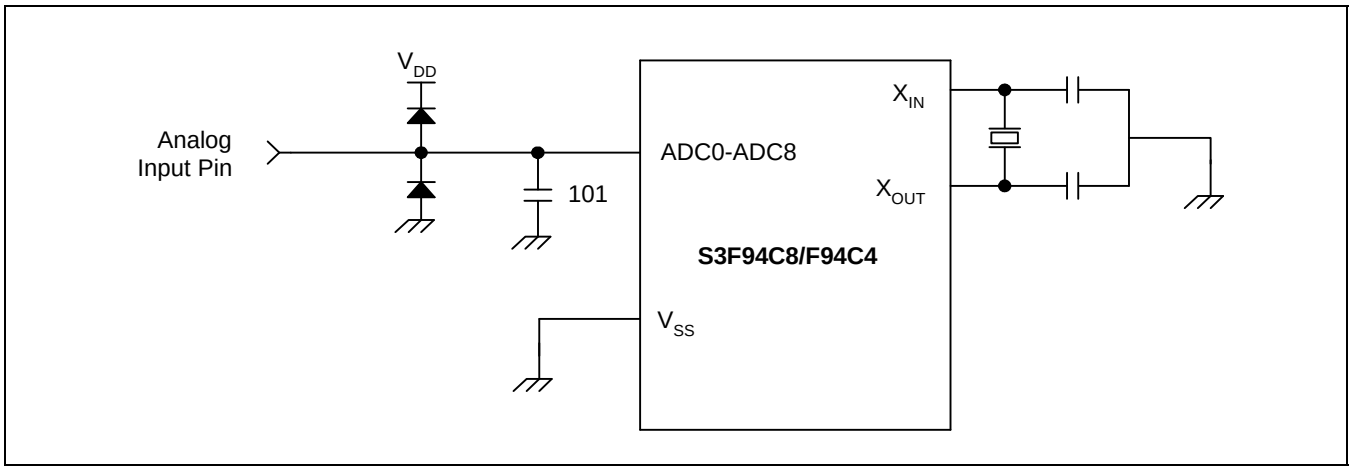


Figure 12-5 Recommended A/D Converter Circuit for Highest Absolute Accuracy

Example 12-1 Programming Tip — Configuring A/D Converter

```
;-----------------<< Interrupt Vector Address >>
            VECTOR 00H, INT_TIMER0     ; S3F94C8/F94C4 has only one interrupt vector
;--------------<< Smart Option >>
            ORG    003CH
            DB     000H                ; 003CH, must be initialized to 0
            DB     000H                ; 003DH, must be initialized to 0
            DB     0FFH                ; 003EH, enable LVR
            DB     003H                ; 003FH, internal RC oscillator
            ORG    0100H
RESET:      DI                         ; Disable Interrupt
            LD     BTCON,#10100011B    ; Disable Watchdog
            •
            •
            •
            LD     P0CONH,#11111111B   ; Configure P0.4–P0.7 AD input
            LD     P0CONL,#11111111B   ; Configure P0.0–P0.3 AD input
            LD     P2CONH,#00100000B   ; Configure P2.6 AD input
            EI                         ; Enable interrupt
;--------------<< Main loop >>
MAIN:
            •
            •
            •
            CALL   AD_CONV             ; Subroutine for AD conversion
            •
            •
            •
            JR     t, MAIN             ;
AD_CONV:
            LD     ADCON,#00000001B    ; Select analog input channel → P0.0
                                       ; Select conversion speed → fOSC/16
                                       ; Set conversion start bit

            NOP
                                       ; If you select conversion speed to fOSC/16
                                       ; at least one NOP must be included
CONV_LOOP:
            TM     ADCON,#00001000B    ; Check EOC flag
            JR     Z,CONV_LOOP         ; If EOC flag = 0, jump to CONV_LOOP until EOC flag
                                               = 1
            LD     R0,ADDATAH          ; High 8-bits of conversion result are stored
                                       ; in ADDATAH register
            LD     R1,ADDATAL          ; Low 2-bits of conversion result are stored
                                       ; in ADDATAL register
            LD     ADCON,#00010011B    ; Select analog input channel → P0.1
                                       ; Select conversion speed → fOSC/8
                                       ; Set conversion start bit
```

```
CONV_LOOP2:
            TM      ADCON,#00001000B    ; Check EOC flag
            JR      Z,CONV_LOOP2
            LD      R2,ADDATAH
            LD      R3,ADDATAL
            •
            •
            •
            RET                         ;
INT_TIMER0:
            •                           ;
            •                           ;
            •                           ; Pending bit clear
            IRET                        ;
            •
            •
            END
```

13 EMBEDDED FLASH MEMORY INTERFACE

13.1 Overview of Embedded Flash Memory Interface

S3F94C8/F94C4 contains an internal on-chip flash memory (instead of masked ROM) that can be accessed by the 'LDC' instruction. This flash memory is sector erasable and byte programmable. You can program the data in a flash memory area any time you want.

The embedded 8K/4K-byte memory in S3F94C8/F94C4 has two operating features, namely:

- Tool Program Mode: For more information, refer to *Chapter 16, "S3F94C8/F94C4 Flash MCU"*.
- User Program Mode

13.1.1 Flash ROM Configuration

The S3F94C8/F94C4 flash memory consists of 64 sectors (S3F94C8) or 32 sectors (S3F94C4). Each sector consists of 128bytes. Therefore, the total size of flash memory is 64 x 128 (8KB) or 32 x 128 bytes (4KB). You can erase the flash memory by a sector unit at a time and write the data into flash memory by a byte unit at a time.

- 8K/4Kbyte Internal flash memory
- Sector size: 128bytes
- 10years data retention
- Fast programming Time:
  Sector Erase: 8ms (minimum)
  Byte Program: 25us (minimum)
- Byte programmable
- User programmable by 'LDC' instruction
- Sector (128-Bytes) erase available
- Endurance: 10,000 Erase/Program cycles (min)

13.1.1.1 Tool Program Mode

This mode is applied at the time of erasing and programming full area of flash memory by using external programming tools. The five pins of S3F94C8/F94C4 are connected to a programming tool. The internal flash memory of S3F94C8/F94C4 can be programmed by using Serial OTP/MTP tools, SPW2 plus single programmer or GW-PRO2 gang programmer, and so on. Other modules (except flash memory module) operate at reset state.

This mode does not support sector erase. However, it supports chip erase (all flash memory is erased at a time) and two protection modes (Hard lock protection/ Read protection). The read protection mode is only available in tool program mode.

To read protect a chip, select read protection option while writing program code to a chip in tool program mode by using a programming tool. After read protect, all data of flash memory read becomes “00”. This protection is released by chip erase execution in tool program mode.

Table 13-1 Descriptions of Pins Used to Read/Write the Flash in Tool Program Mode

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P0.1	SDAT	18 (20-pin) 14 (16-pin)	I/O	Serial data pin (Output when reading, Input when writing) Input and push-pull output port can be assigned
P0.0	SCLK	19 (20-pin) 15 (16-pin)	I	Serial clock pin (input only pin)
RESET/P1.2	V_{PP}	4	I	Power supply pin for Tool mode entering (indicates that MTP enters into the Tool mode). When 11 V is applied, MTP is in Tool mode.
V_{DD}/V_{SS}	V_{DD}/V_{SS}	20 (20-pin), 16 (16-pin) 1 (20-pin), 1 (16-pin)	I	Logic power supply pin.

13.1.1.2 User Program Mode

This mode supports sector erase, byte programming, byte read, and protection mode (Hard Lock Protection). The S3F94C8/F94C4 contains an internal pumping circuit to generate high voltage. To program a flash memory in this mode, several control registers should be used.

There are four kinds of functions in user program mode – programming, reading, sector erase, and protection mode (Hard lock protection).

13.2 Flash Memory Control Registers (User Program Mode)

13.2.1 Flash Memory Control Register (FMCOn)

FMCON register is only available in the user program mode to select the flash memory operation mode: sector erase, byte programming, or making the flash memory into a hard lock protection.

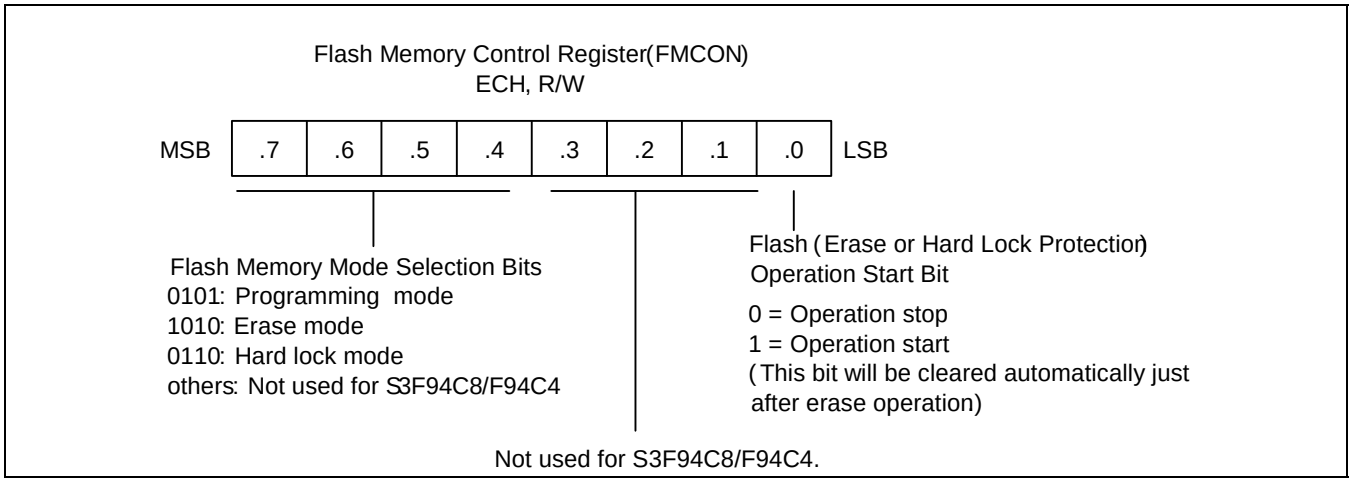


Figure 13-1 Flash Memory Control Register (FMCON)

The bit 0 of FMCON register (FMCON.0) is used for enabling the Erase and Hard Lock Protection. Therefore, Erase and Hard Lock Protection is activated when you set FMCON.0 to “1”. If you write FMCON.0 to 1 for erasing, CPU is stopped automatically for erasing time (minimum 4ms). After erasing time finishes, CPU is restarted automatically. When you read or program a byte data from (or into) flash memory, this bit is not changed.

13.2.2 Flash Memory User Programming Enable Register (FMUSR)

The FMUSR register is used for safe operation of the flash memory. This register will protect undesired erase or malfunctioning of CPU caused by an electrical noise. After reset, the user program mode is disabled, since the value of FMUSR is “00000000B” by reset operation. If it is necessary to operate the flash memory, you can use the user program mode by setting the value of FMUSR to “10100101B”. User program mode is disabled for the other value of “10100101B”.

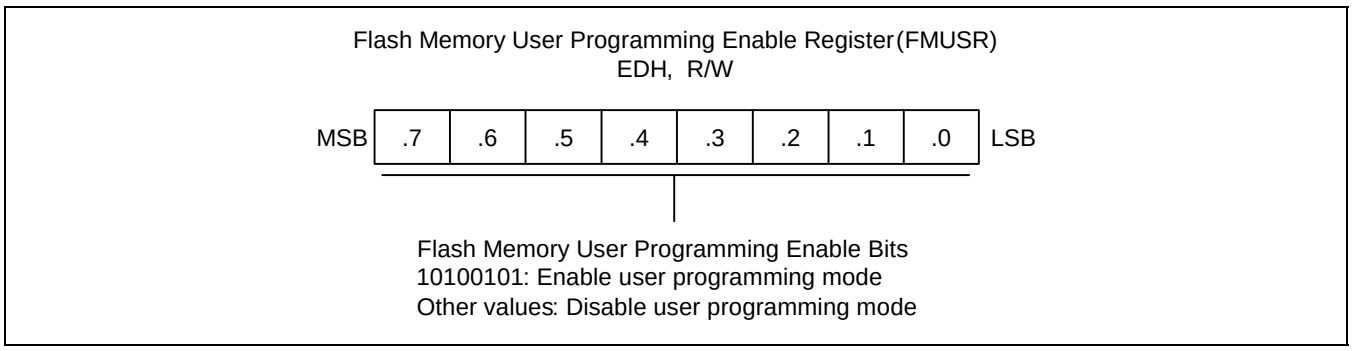


Figure 13-2 Flash Memory User Programming Enable Register (FMUSR)

13.2.3 Flash Memory Sector Address Registers

There are two sector address registers for erase or programming flash memory. The Flash Memory Sector Address Register Low Byte (FMSECL) indicates the low byte of sector address and Flash Memory Address Sector Register High Byte (FMSECH) indicates the high byte of sector address. FMSECH is needed for S3F94C8/F94C4 since it has 64/32 sectors.

One sector consists of 128 bytes. Each sector's address starts with XX00H or XX80H, that is, a base address of sector is XX00H or XX80H. Therefore, bit .6-.0 of FMSECL does not mean that the value is '1' or '0'. The simplest way is to load the sector base address into FMSECH and FMSECL register.

When programming the flash memory, user should program after loading a sector base address, which is located in the destination address to write data into FMSECH and FMSECL register. If the next operation is also to write one byte data, user should check whether next destination address is located in the same sector or not.. In case of other sectors, load the sector address to FMSECH and FMSECL registers according to the sector. (Refer to Pages 13-11 *Programming Tip — Programming for more information*.)

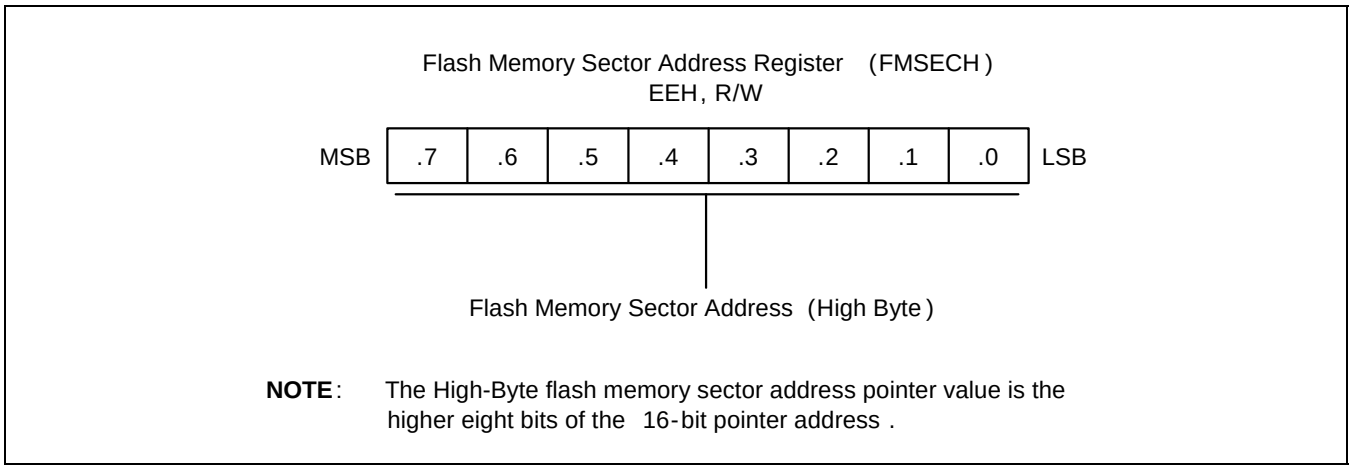


Figure 13-3 Flash Memory Sector Address Register (FMSECH)

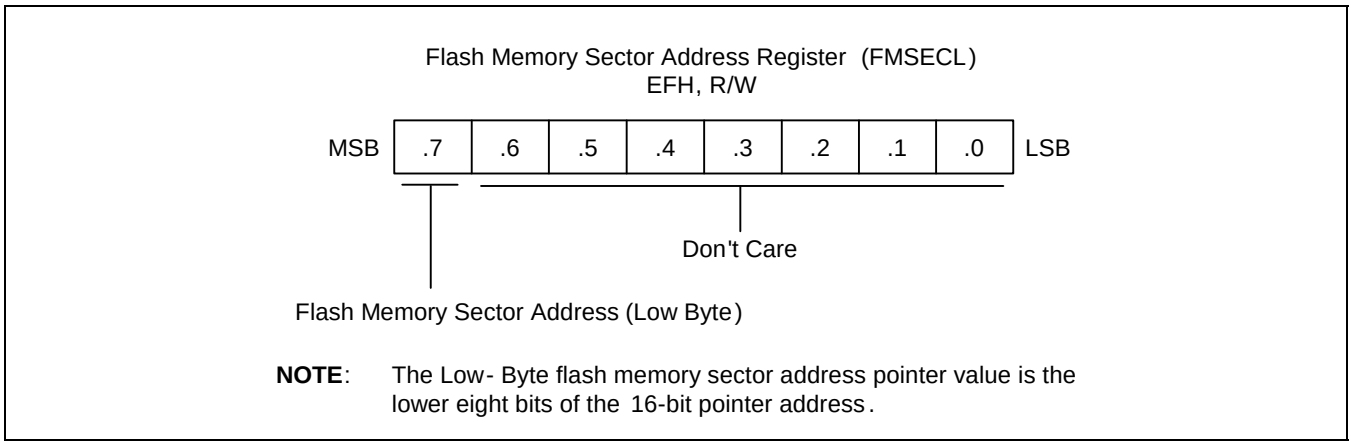


Figure 13-4 Flash Memory Sector Address Register (FMSECL)

13.3 Sector Erase

You can erase a flash memory partially by using sector erase function (only in user program mode). The unit of flash memory that can be erased in the user program mode is a sector. The program memory of S3F94C8/F94C4 8K/4Kbytes flash memory is divided into 64/32 sectors. Every sector has 128 bytes size in all. Therefore, the sector to be located in destination address should be erased first to program a new data (one byte) into flash memory. Minimum 4ms delay time for the erase is required after setting the sector address and triggering erase start bit (FMCON.0). Sector erase is not supported in tool program modes (MDS mode tool or programming tool).

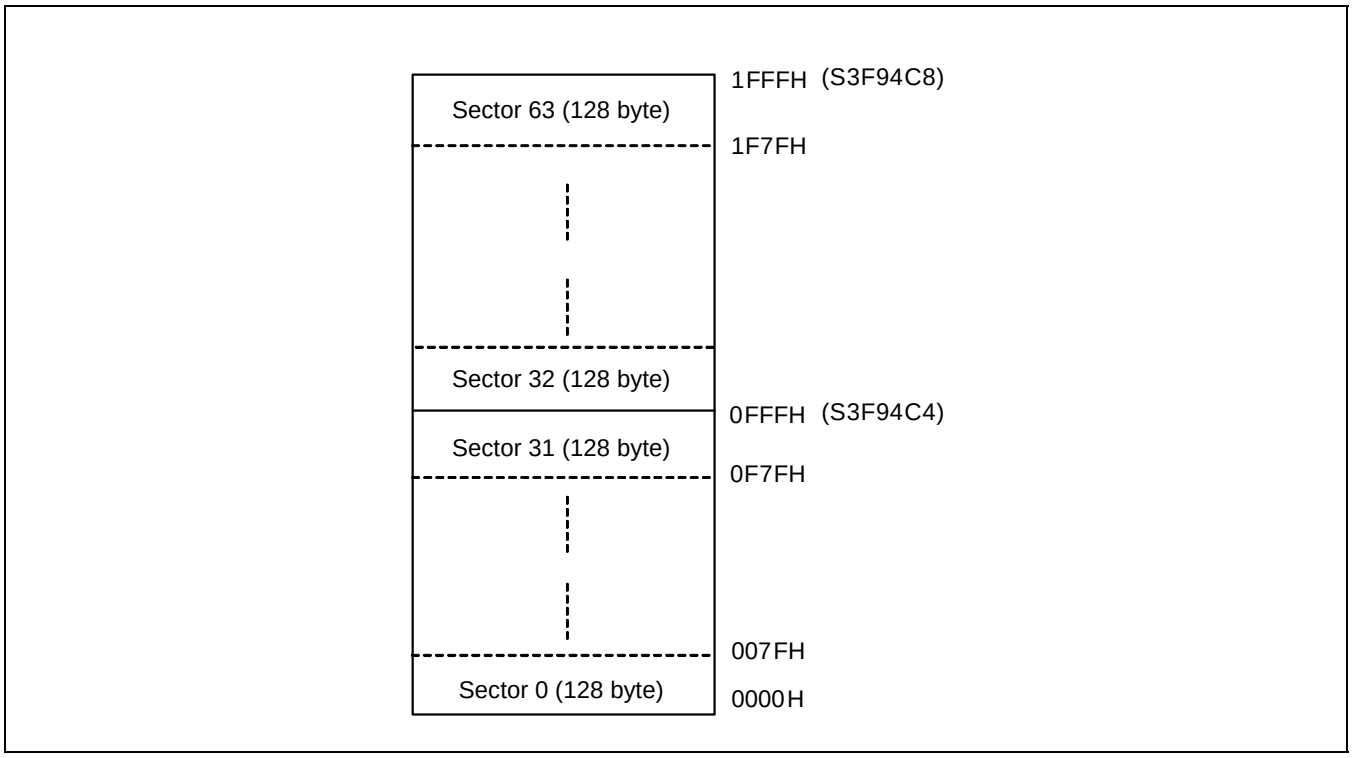


Figure 13-5 Sector configurations in User Program Mode

13.3.1 The Sector Erase Procedure in User Program Mode

1. Set Flash Memory User Programming Enable Register (FMUSR) to “10100101B”.
2. Set Flash Memory Sector Address Register (FMSECH and FMSECL).
3. Set Flash Memory Control Register (FMCON) to “10100001B”.
4. Set Flash Memory User Programming Enable Register (FMUSR) to “00000000B”.

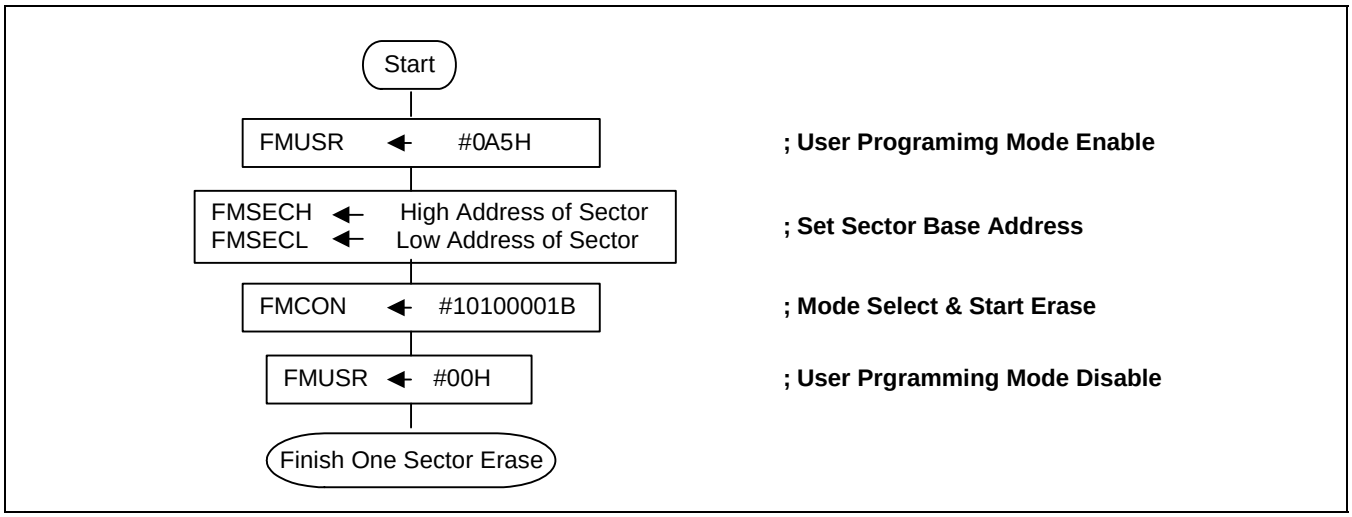

Start
FMUSR ← #0A5H
; User Programimg Mode Enable
FMSECH ← High Address of Sector
FMSECL ← Low Address of Sector
; Set Sector Base Address
FMCON ← #10100001B
; Mode Select & Start Erase
FMUSR ← #00H
; User Prgramming Mode Disable
Finish One Sector Erase

Figure 13-6 Sector Erase Flowchart in User Program Mode

NOTE:

1. If you erase a sector selected by Flash Memory Sector Address Registers FMSECH and FMSECL, FMUSR should be enabled just before starting the sector erase operation. To erase a sector, Flash Operation Start Bit of FMCON register is written from operation stop ‘0’ to operation start ‘1’. This bit will be cleared automatically just after the corresponding operation is completed. In other words, when S3F94C8/F94C4 is in a condition where flash memory user programming enable bits are enabled and start operation of sector erase is executed, it will get the result of erasing selected sector as user’s purpose. The Flash Operation Start Bit of FMCON register is also clear automatically.
2. If you execute sector erase operation with FMUSR disabled, then Flash Operation Start Bit, FMCON.0 bit, remains high (which means start operation), and is not cleared even though next instruction is executed. Therefore, you should be careful to set FMUSR when executing sector erase for no effect on other flash sectors.

Example 13-1 Programming Tip — Sector Erase

Case1. Erase one sector

```
      •
      •
ERASE_ONESECTOR:
      LD     FMUSR,#0A5H          ; Enable User program mode
      LD     FMSECH,#04H          ; Set sector address 0400H, sector 8,
      LD     FMSECL,#00H          ; among sector 0~32
      LD     FMCON,#10100001B     ; Select erase mode enable and Start sector erase
ERASE_STOP:
      LD     FMUSR,#00H           ; Disable User program mode
```

13.4 Programming

A flash memory is programmed in one-byte unit after sector erase. The write operation of programming starts by ‘LDC’ instruction.

13.4.1 Program Procedure in User Program Mode

1. Erase target sectors before programming.
2. Set Flash Memory User Programming Enable Register (FMUSR) to “10100101B”.
3. Set Flash Memory Control Register (FMCON) to “0101000XB”.
4. To write data, set Flash Memory Sector Address Register (FMSECH and FMSECL) to the sector base address of destination address.
5. Load transmission data into a working register.
6. Load flash memory upper address into upper register of pair working register.
7. Load flash memory lower address into lower register of pair working register.
8. Load transmission data to flash memory location area on ‘LDC’ instruction by indirectly addressing mode.
9. Set Flash Memory User Programming Enable Register (FMUSR) to “00000000B”.

NOTE: In programming mode, it does not matter whether FMCON.0’s value is “0” or “1”.

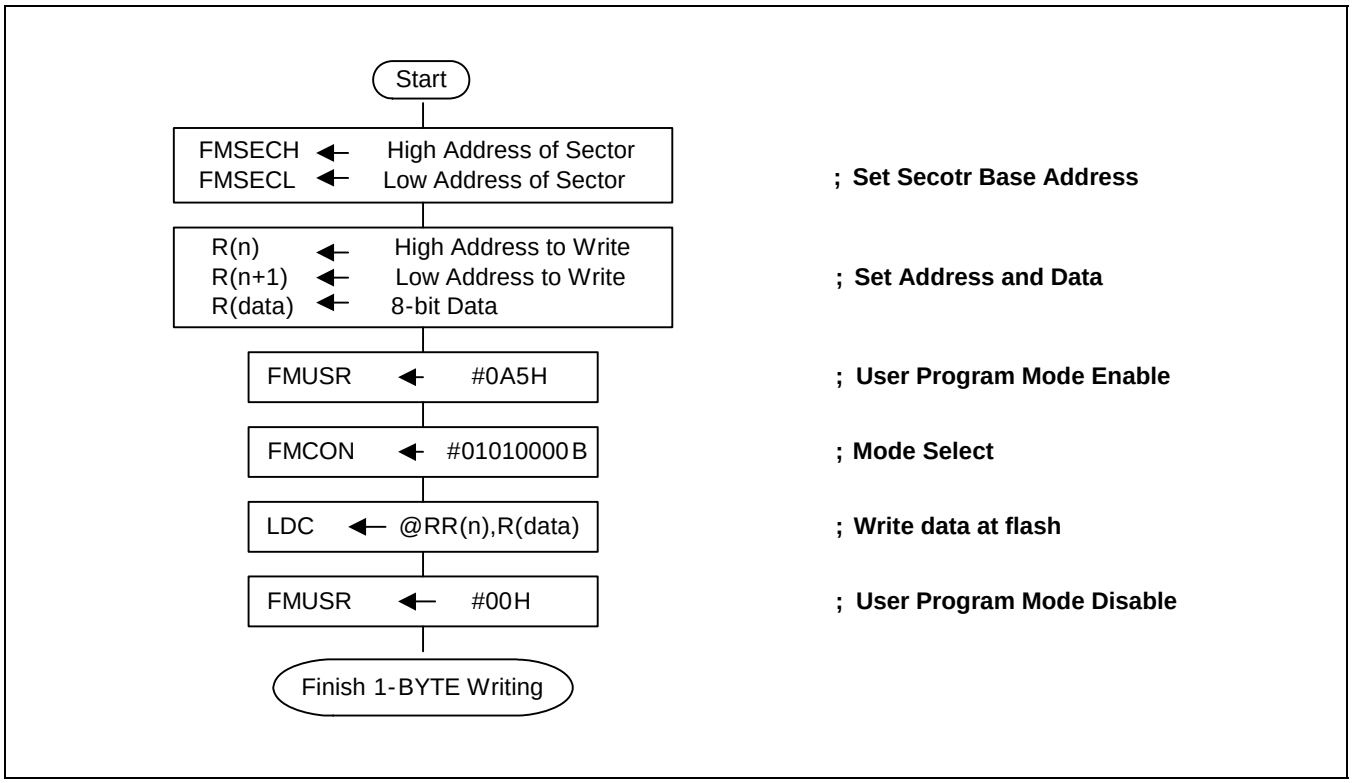

Start
FMSECH ← High Address of Sector
FMSECL ← Low Address of Sector
; Set Secotr Base Address
R(n) ← High Address to Write
R(n+1) ← Low Address to Write
R(data) ← 8-bit Data
; Set Address and Data
FMUSR ← #0A5H
; User Program Mode Enable
FMCON ← #01010000B
; Mode Select
LDC ← @RR(n),R(data)
; Write data at flash
FMUSR ← #00H
; User Program Mode Disable
Finish 1-BYTE Writing

Figure 13-7 Byte Program Flowchart in a User Program Mode

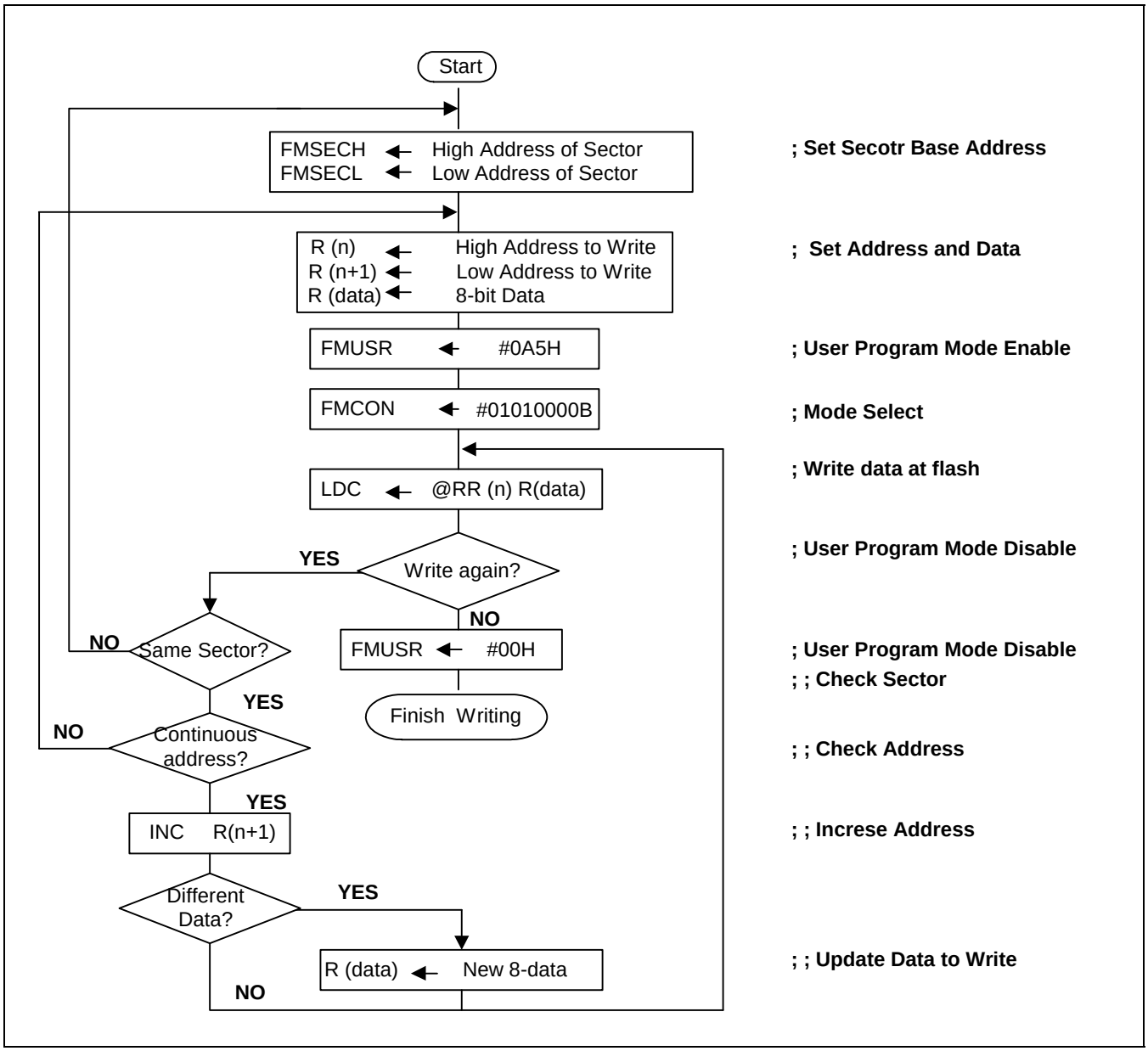


Figure 13-8 Program Flowchart in a User Program Mode

Example 13-2 Programming Tip — Programming

Case1. 1-Byte Programming

```
WR_BYTE:                        ; Write data “AAH”  to destination address 0310H
      LD     FMUSR,#0A5H        ; Enable User program mode
      LD     FMCON,#01010000B   ; Select Programming mode
      LD     FMSECH, #03H       ; Set the base address of sector (0300H)
      LD     FMSECL, #00H
      LD     R9,#0AAH           ; Load data “AA” to write
      LD     R10,#03H           ; Load flash memory upper address into upper register of
      `                           pair working
                                ; register
      LD     R11,#10H           ; Load flash memory lower address into lower register of
      `                           pair working
                                ; register
      LDC    @RR10,R9           ; Write data ‘AAH’ at flash memory location (0310H)
      LD     FMUSR,#00H         ; Disable User program mode
```

Case2. Programming in the same sector

```
WR_INSECTOR:                    ; RR10-->Address copy (R10–high address,R11-low address)
      LD     R0,#40H

      LD     FMUSR,#0A5H        ; Enable User program mode
      LD     FMCON,#01010000B   ; Select Programming mode and start programming
      LD     FMSECH,#06H        ; Set the base address of sector located in target
                                  to write data
      LD     FMSECL,#00H        ; The sector 12’s base address is 0600H.
      LD     R9,#33H            ; Load data “33H” to write
      LD     R10,#06H           ; Load flash memory upper address into upper register
                                  pair working
                                ; register
      LD     R11,#00H           ; Load flash memory lower address into lower register of
                                  pair working
                                ; register
WR_BYTE:
      LDC    @RR10,R9           ; Write data ‘33H’ at flash memory location
      INC    R11                ; Reset address in the same sector by INC instruction
      DEC    R0
      JP     NZ, WR_BYTE        ; Check whether the end address for programming reaches
                                  0640H.
      LD     FMUSR,#00H         ; Disable User Program mode
```

Case3. Programming to the flash memory space located in other sectors

```
      •
      •
WR_INSECTOR2:
      LD     R0,#40H
      LD     R1,#40H
      LD     FMUSR,#0A5H          ; Enable User program mode
      LD     FMCON,#01010000B     ; Select programming mode and start programming
      LD     FMSECH,#01H          ; Set the base address of sector located in target address
                                    to write data
      LD     FMSECL,#00H          ; The sector 2’s base address is 100H
      LD     R9,#0CCH             ; Load data “CCH” to write
      LD     R10,#01H             ; Load flash memory upper address into upper register of
                                    pair working
                                  ; register
      LD     R11,#40H             ; Load flash memory lower address into lower register of
                                    pair working
                                  ; register
      CALL   WR_BYTE

      LD     R0,#40H
WR_INSECTOR5:
      LD     FMSECH,#02H          ; Set the base address of sector located in target address
                                    to write data
      LD     FMSECL,#80H          ; The sector 5’s base address is 0280H
      LD     R9,# 55H             ; Load data “55H” to write
      LD     R10,#02H             ; Load  flash memory upper address into upper register of
                                    pair working
                                  ; register
      LD     R11,#90H             ; Load flash memory lower address into lower register of
                                    pair working
                                  ; register
      CALL   WR_BYTE
WR_INSECTOR12:
      LD     FMSECH,#06H          ; Set the base address of sector located target address to
                                    write data
      LD     FMSECL,#00H          ; The sector 12’s base address is 0600H
      LD     R9,#0A3H             ; Load data “A3H” to write
      LD     R10,#06H             ; Load  flash memory upper address into upper register of
                                    pair working
                                  ; register
      LD     R11,#40H             ; Load flash memory lower address into lower register of
                                    pair working
                                  ; register
```

```
WR_BYTE1:
      LDC    @RR10,R9            ; Write data ‘A3H’ at flash memory location
      INC    R11
      DEC    R1
      JP     NZ, WR_BYTE1
      LD     FMUSR,#00H          ; Disable User Program mode
      •
      •
WR_BYTE:
      LDC    @RR10,R9            ; Write data written by R9  at flash  memory location
      INC    R11
      DEC    R0
      JP     NZ, WR_BYTE
      RET
```

13.5 Reading

The read operation starts by ‘LDC’ instruction.

13.5.1 Program Procedure in User Program Mode

1. Load a flash memory upper address into upper register of pair working register.
2. Load a flash memory lower address into lower register of pair working register.
3. Load received data from flash memory location area on ‘LDC’ instruction by indirectly addressing mode.

Example 13-3 Programming Tip — Reading

```
      •
      •
      LD     R2,#03H             ; Load flash memory’s upper address
                                 ; to upper register of pair working register
      LD     R3,#00H             ; Load flash memory’s lower address
                                 ; to lower register of pair working register
LOOP:
      LDC    R0,@RR2             ; Read data from flash memory location
                                 ; (Between 300H and 3FFH)
      INC    R3
      CP     R3,#0FFH
      JP     NZ,LOOP
      •
      •
      •
      •
```

13.6 Hard Lock Protection

You can set Hard Lock Protection by writing ‘0110B’ in FMCON7-4. This function prevents change of data in flash memory area. If this function is enabled, you cannot write or erase the data in a flash memory area. This protection can be released by executing chip erase in the tool program mode.

In user program mode, the procedure of setting Hard Lock Protection is described below.In tool mode, the manufacturer of serial tool writer could support Hardware Protection. Refer to the manual of serial program writer tool provided by the manufacturer.

13.6.1 Program Procedure in User Program Mode

1. Set Flash Memory User Programming Enable Register (FMUSR) to “10100101B”.
2. Set Flash Memory Control Register (FMCON) to “01100001B”.
3. Set Flash Memory User Programming Enable Register (FMUSR) to “00000000B”.

Example 13-4 Programming Tip — Hard Lock Protection

```
LD     FMUSR,#0A5H          ; Enable User program mode
LD     FMCON,#01100001B     ; Select Hard Lock Mode and start protection
LD     FMUSR,#00H           ; Disable User program mode
•
•
```

14 ELECTRICAL DATA

14.1 Overview of Electrical Data

In this section, the following S3F94C8/F94C4 electrical characteristics have been presented in the form of tables and graphs:

- Absolute maximum ratings
- DC electrical characteristics
- AC electrical characteristics
- Input timing measurement points
- Oscillator characteristics
- Oscillation stabilization time
- Operating voltage range
- Schmitt trigger input characteristics
- Data retention supply voltage in Stop mode
- Stop mode release timing when initiated by a RESET
- A/D converter electrical characteristics
- LVR circuit characteristics
- LVR reset timing
- Full Flash Memory Characteristics
- ESD Characteristics

Table 14-1 Absolute Maximum Ratings

(TA = 25°C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 6.5	V
Input voltage	V_I	All ports	– 0.3 to V_{DD} + 0.3	V
Output voltage	V_O	All output ports	– 0.3 to V_{DD} + 0.3	V
Output current high	I_{OH}	One I/O pin is active	– 25	mA
		All I/O pins are active	– 80	
Output current low	I_{OL}	One I/O pin is active	+ 30	mA
		All I/O pins are active	+ 100	–
Operating temperature	T_A	–	– 40 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	°C

Table 14-2 DC Electrical Characteristics

(TA = – 40 °C to + 85 °C, VDD = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
Operating voltage	V_{DD}	fmain=0.4 – 4 MHz		1.8	–	5.5	V
		fmain=0.4 – 10 MHz		2.7	–	5.5	
Main crystal or ceramic frequency	f_{main}	V_{DD} = 2.7 V to 5.5V		0.4	–	10	MHz
		V_{DD} = 1.8 V to 2.7V		0.4	–	4	
Input high voltage	V_{IH1}	Ports 0,1, 2, and RESET	V_{DD} = 1.8 to 5.5V	0.8 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN} and X_{OUT}		V_{DD} - 0.1			
Input low voltage	V_{IL1}	Ports 0, 1, 2, and RESET	V_{DD} = 1.8 to 5.5V	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN} and X_{OUT}				0.1	
Output high voltage	V_{OH}	I_{OH} = – 10 mA Ports 0, 2, and P1.0-P1.1	V_{DD} = 4.5 to 5.5V	V_{DD} -1.5	V_{DD} - 0.4	–	V
Output low voltage	V_{OL}	I_{OL} = 25 mA Ports 0, 2, and P1.0-P1.1	V_{DD} = 4.5 to 5.5V	–	0.4	2.0	V
Input high leakage current	I_{LIH1}	All input pins (except $I_{LIH2,P1.2}$[2])	V_{IN} = V_{DD}	–	–	1	uA
	I_{LIH2}	X_{IN}	V_{IN} = V_{DD}			20	
Input low leakage current	I_{LIL1}	All input pins (except I_{LIL2})	V_{IN} = 0 V	–	–	–1	uA
	I_{LIL2}	X_{IN}	V_{IN} = 0 V			–20	
Output high leakage current	I_{LOH}	All output pins	V_{OUT} = V_{DD}	–	–	2	uA
Output low leakage current	I_{LOL}	All output pins	V_{OUT} = 0 V	–	–	–2	uA
Pull-up resistors	R_{P1}	V_{IN} = 0 V, Ports 0, 2, and P1.0-P1.1	V_{DD} = 5 V T_A=25°C	25	50	100	kΩ
Pull-down resistors	R_{P2}	V_{IN} = 0 V, P1.0-P1.1[2]	V_{DD} = 5 V T_A=25°C	25	50	100	
Supply current (1)	I_{DD1}	Run mode 10 MHz CPU clock	V_{DD} = 4.5 to 5.5V	–	2	5	mA
		3MHz CPU clock	V_{DD} = 2.0V	–	1	2	
	I_{DD2}	Idle mode 10 MHz CPU clock	V_{DD} = 4.5 to 5.5V	–	1.5	3.0	
		3MHz CPU clock	V_{DD} = 2.0V	–	0.5	1.5	

Parameter	Symbol	Conditions		Min	Typ	Max	Unit
	I_{DD3}	Stop mode	V_{DD} = 4.5 to 5.5V (LVR disable) T_A = 25 °C	–	0.3	2.0	uA
			V_{DD} = 4.5 to 5.5V (LVR disable) T_A= – 40°C to +85°C	–	1.0	4.0	
			V_{DD} = 4.5 to 5.5V (LVR enable) T_A = 25 °C	–	40	80	
			V_{DD} = 2.6 V (LVR enable) T_A = 25 °C	–	30	60	

NOTE:

1. Supply current does not include the current drawn through internal pull-up resistors or external output current loads and ADC module.
2. P1.2 has intrinsic internal pull-down resistor (Internal VPP circuit), whose typical value is about 300Kohm.

Table 14-3 AC Electrical Characteristics

(TA = – 40 °C to + 85°C, VDD = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input high, low width	t_{INTH} t_{INTL}	INT0, INT1 V_{DD} = 5 V ± 10 %	–	200	–	ns
RESET input low width	t_{RSL}	Input V_{DD} = 5 V ± 10 %	1	–	–	us

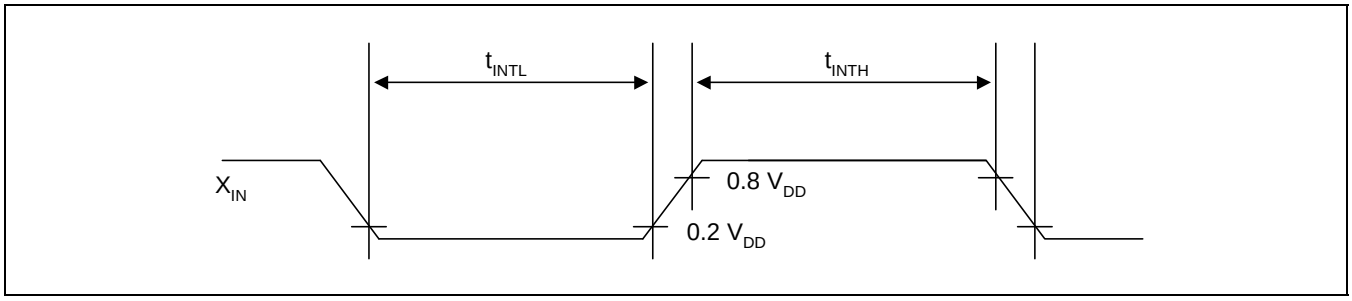
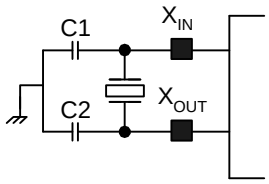
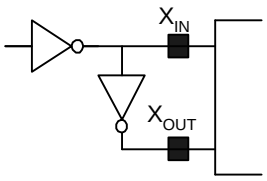


Figure 14-1 Input Timing Measurement Points

Table 14-4 Oscillator Characteristics

(TA = – 40°C to + 85 °C)

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
Main crystal or ceramic		V_{DD} = 2.7 to 5.5 V	0.4	–	10	MHz
		V_{DD1} = 1.8 to 2.7 V	0.4	–	4	MHz
External clock (Main System)		V_{DD} = 2.7 to 5.5 V	0.4	–	10	MHz
		V_{DD} = 1.8 to 2.7 V	0.4	–	4	MHz

NOTE: For more information, refer to *Figure 14-2*.

Table 14-5 Oscillation Stabilization Time

(TA = - 40 °C to + 85 °C, VDD = 1.8 V to 5.5 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Main crystal	f_{OSC} > 1.0 MHz Oscillation stabilization occurs when V_{DD} is equal to the minimum oscillator voltage range.	–	–	20	ms
Main ceramic		–	–	10	ms
External clock (main system)	X_{IN} input high and low width (t_{XH}, t_{XL})	25	–	500	ns
Oscillator stabilization wait time	t_{WAIT} when released by a reset (1)	–	$2^{19}/f_{OSC}$	–	ms
	t_{WAIT} when released by an interrupt (2)	–	–	–	ms

NOTE:

1. fOSC specifies the oscillator frequency.
2. The duration of the oscillator stabilization wait time, tWAIT, when it is released by an interrupt is determined by the settings in the basic timer control register, BTCON.

Table 14-6 RC Oscillator Characteristics (S3F94C8EZZ / F94C4EZZ)

(TA = – 25 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
External RC oscillator	–	V_{DD} = 5 V	–	4	–	MHz
Internal RC oscillator	–	–	–	3.2	–	MHz
				500	–	KHz
Tolerance of Internal RC	–	V_{DD} = 5.0 V TA = 25 °C	–	–	±3	%
		V_{DD} = 5.0 V TA = – 25°C to + 85 °C	–	–	±5	%
		V_{DD} = 2.0 to 5.5 V TA = – 25°C to + 85 °C	–	–	±3	%

Table 14-7 RC Oscillator Characteristics (S3F94C8XZZ / F94C4XZZ)

(TA = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Oscillator	Clock Circuit	Test Condition	Min	Typ	Max	Unit
External RC oscillator	–	V_{DD} = 5 V	–	4	–	MHz
Internal RC oscillator	–	–	–	3.2	–	MHz
				500	–	KHz
Tolerance of Internal RC	–	V_{DD} = 5.0 V TA = 25 °C	–	–	±1 (TBD)	%
		V_{DD} = 5.0 V TA = – 40°C to + 85 °C	–	–	±3 (TBD)	%
		V_{DD} = 2.0 to 5.5 V TA = – 40°C to + 85 °C	–	–	±5 (TBD)	%

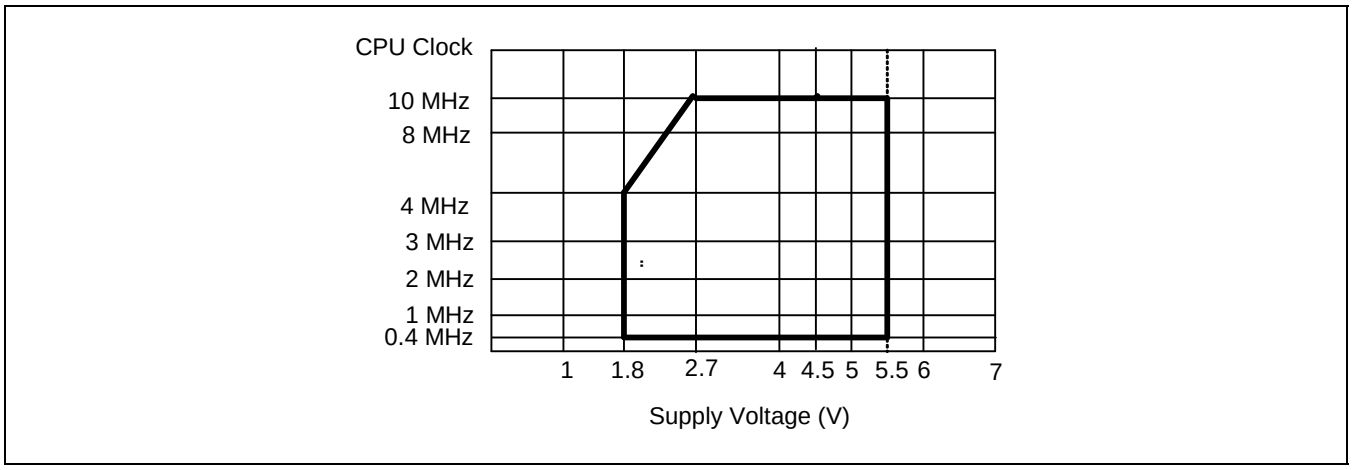


Figure 14-2 Operating Voltage Range

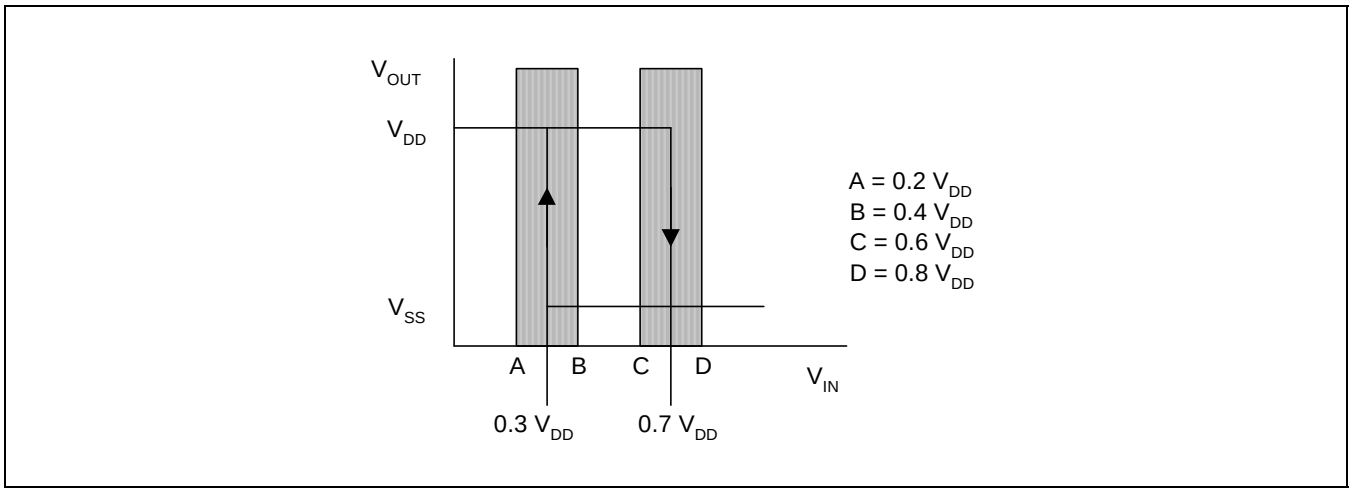


Figure 14-3 Schmitt Trigger Input Characteristics Diagram

Table 14-8 Data Retention Supply Voltage in Stop Mode

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	Stop mode	1.0	–	5.5	V
Data retention supply current	I_{DDDR}	Stop mode; V_{DDDR} = 2.0 V	–	–	1	uA

NOTE: Supply current does not include current drawn through internal pull-up resistors or external output current loads.

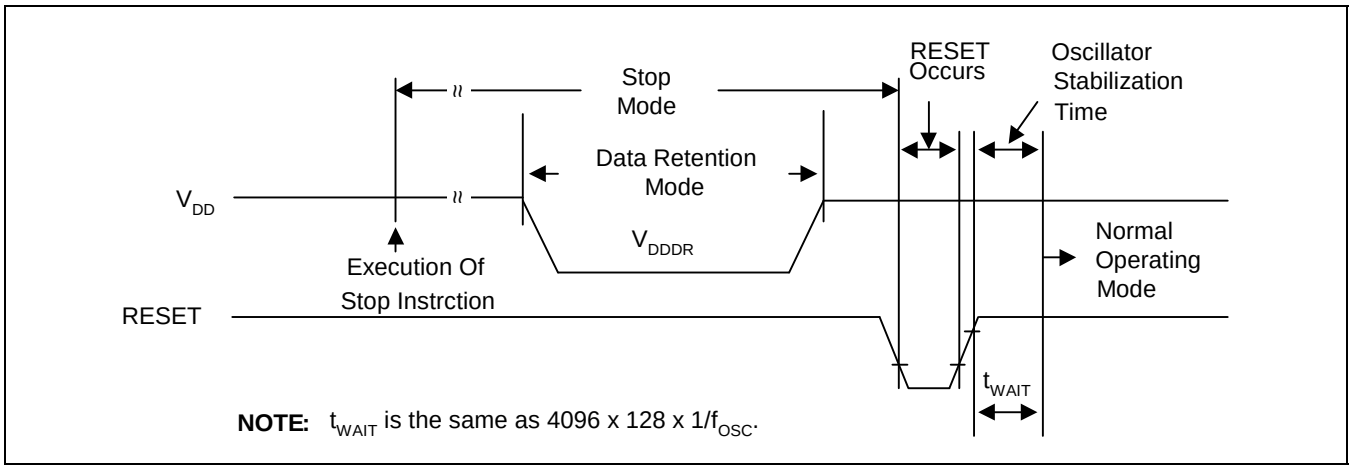


Figure 14-4 Stop Mode Release Timing When Initiated by a RESET

Table 14-9 A/D Converter Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Resolution	–	–	–	10	–	bit
Total accuracy	–	V_{DD} = 5.0 V	–	–	± 3	LSB
Integral linearity error	ILE	V_{DD} = 5.00 V CPU clock = 10 MHz V_{SS} = 0 V	–	–	± 2	LSB
Differential linearity error	DLE	"	–	–	± 1	LSB
Offset error of top	EOT	"	–	± 1	± 3	LSB
Offset error of bottom	EOB	"	–	± 1	± 3	LSB
Conversion time (1)	t_{CON}	–	12.5	20	–	µs
Analog input voltage	V_{IAN}	–	V_{SS}	–	V_{DD}	V
Analog input impedance	R_{AN}	–	2	1000	–	MΩ
Analog input current	I_{ADIN}	V_{DD} = 5 V	–	–	10	µA
Analog block current (2)	I_{ADC}	V_{DD} = 5 V	–	0.5	1.5	mA
		V_{DD} = 3 V	–	0.15	0.45	mA
		V_{DD} = 5 V power down mode	–	100	500	nA

NOTE:

1. "Conversion time" is the time required from the moment a conversion operation starts until it ends.
2. I_{ADC} specifies the operating current during A/D conversion.
3. f_{OSC} specifies the main oscillator clock.

Table 14-10 LVR Circuit Characteristics

(TA = – 40 °C to + 85 °C, V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Low voltage reset	V_{LVR}	–	1.8 2.0 2.7 3.3 3.6	1.9 2.3 3.0 3.6 3.9	2.0 2.6 3.3 3.9 4.2	V

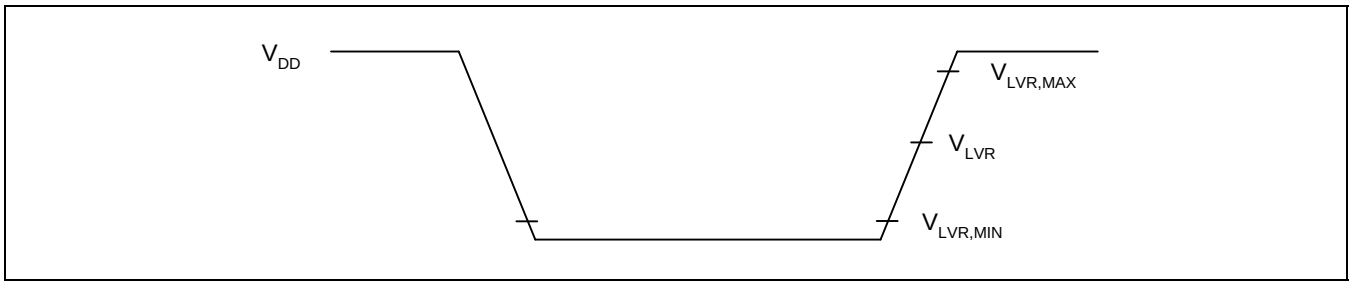


Figure 14-5 LVR Reset Timing

Table 14-11 Flash Memory AC Electrical characteristics

(T_A = – 40 °C to + 85 °C at V_{DD} = 1.8 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Flash Erase/Write/Read Voltage	Fewrv	V_{DD}	1.8	5.0	5.5	V
Programming Time (1)	Ftp	–	20	–	30	uS
Chip Erasing Time (2)	Ftp1	–	32	–	70	mS
Sector Erasing Time (3)	Ftp2	–	4	–	12	mS
Data Access Time	Ft_{RS}	V_{DD} = 2.0V	–	250	–	nS
Number of writing/erasing	FNwe	–	10,000	–	–	Times
Data Retention	Ftdr	–	10	–	–	Years

NOTE:

1. The programming time specifies the time during which one byte (8-bit) is programmed.
2. The Chip erasing time specifies the time during which the entire program memory is erased.
3. The Sector erasing time specifies the time during which all the 128-byte blocks are erased.
4. The chip erasing is available in Tool Program Mode only.

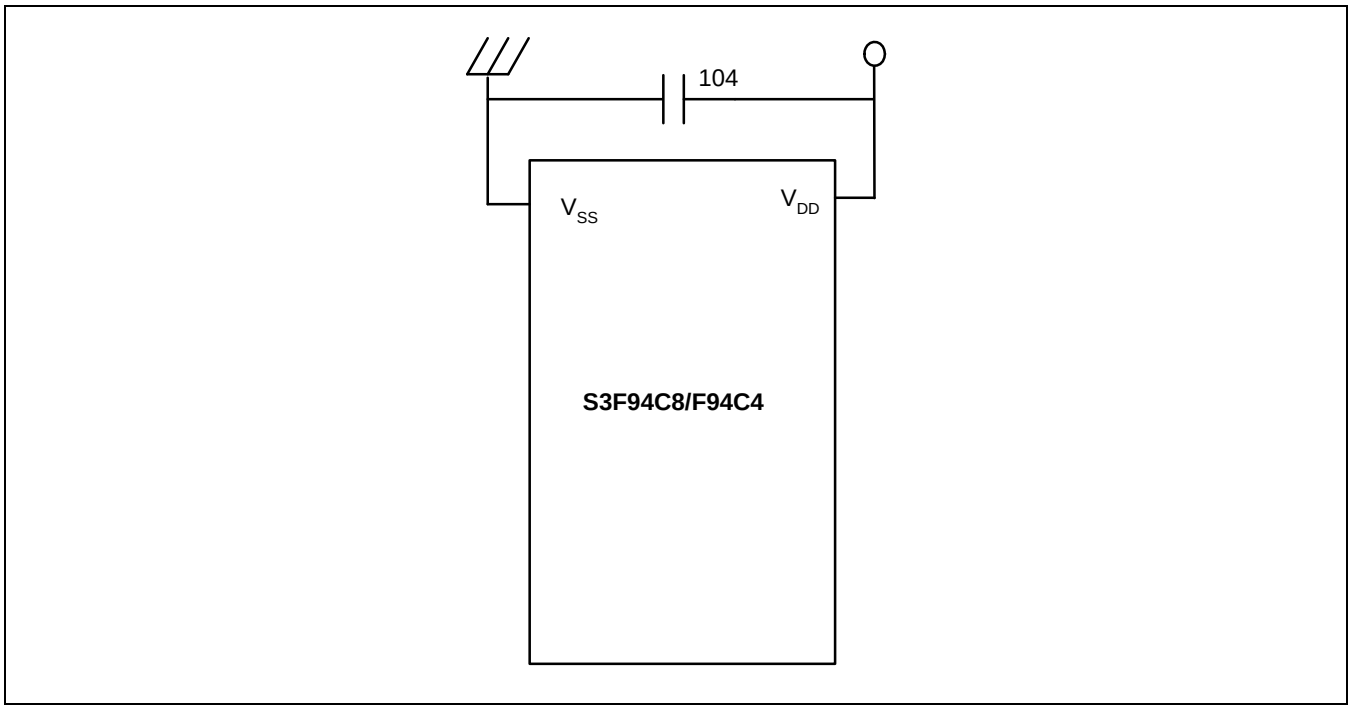


Figure 14-6 Circuit Diagram to Improve EFT Characteristics

NOTE: To improve the EFT characteristics, it is recommended to use the power capacitor near S3F94C8/F94C4 (as shown in *Figure 14-6*).

Table 14-12 ESD Characteristics

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Electrostatic discharge	V_{ESD}	HBM	2000	–	–	V
		MM	200	–	–	V
		CDM	500	–	–	V

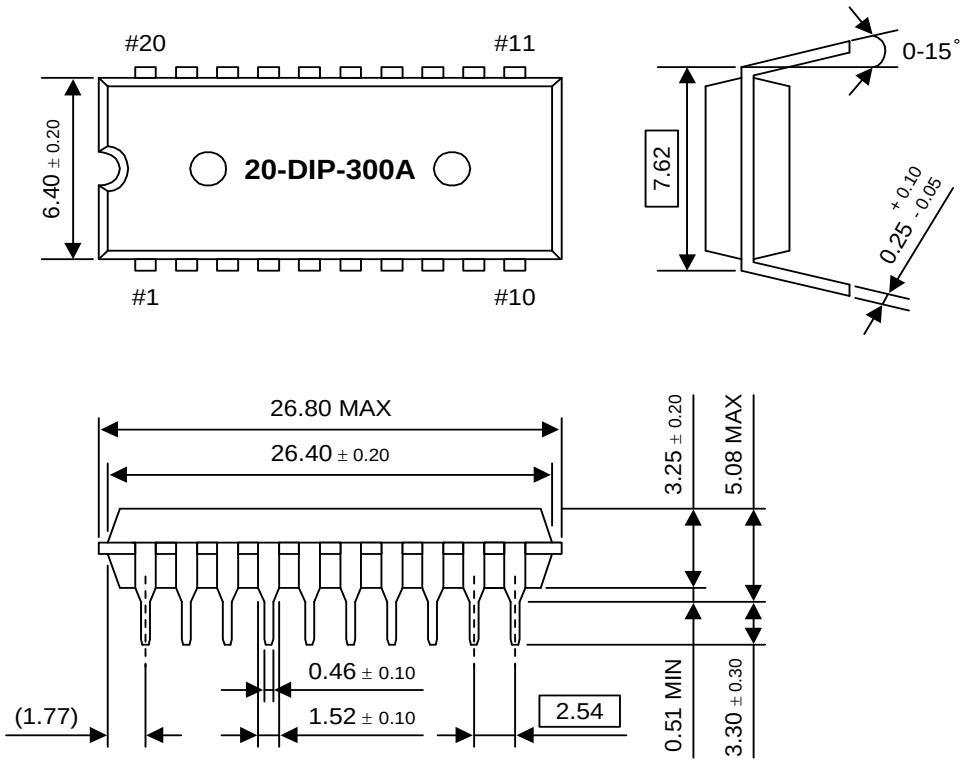
15 MECHANICAL DATA

15.1 Overview of Mechanical Data

S3F94C8/F94C4 is available in the following package:

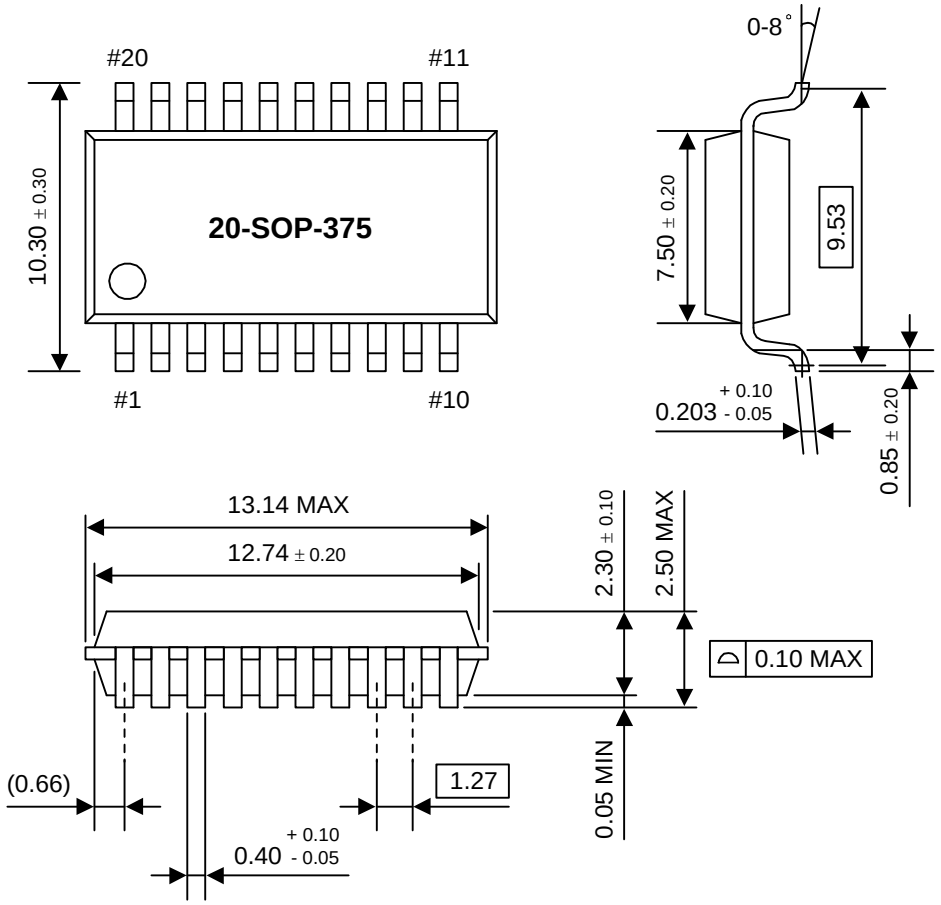
- 20-pin DIP package (Samsung: 20-DIP-300A)
- 20-pin SOP package (Samsung: 20-SOP-375)
- 20-pin SSOP package (Samsung: 20-SSOP-225)
- 16-pin SOP package (Samsung: 16-SOP-225)
- 16-pin TSSOP package (Samsung: 16-TSSOP-0044)

Figure 15-1, *Figure 15-2*, *Figure 15-3*, *Figure 15-4*, and *Figure 15-5* show the dimensions of each package.



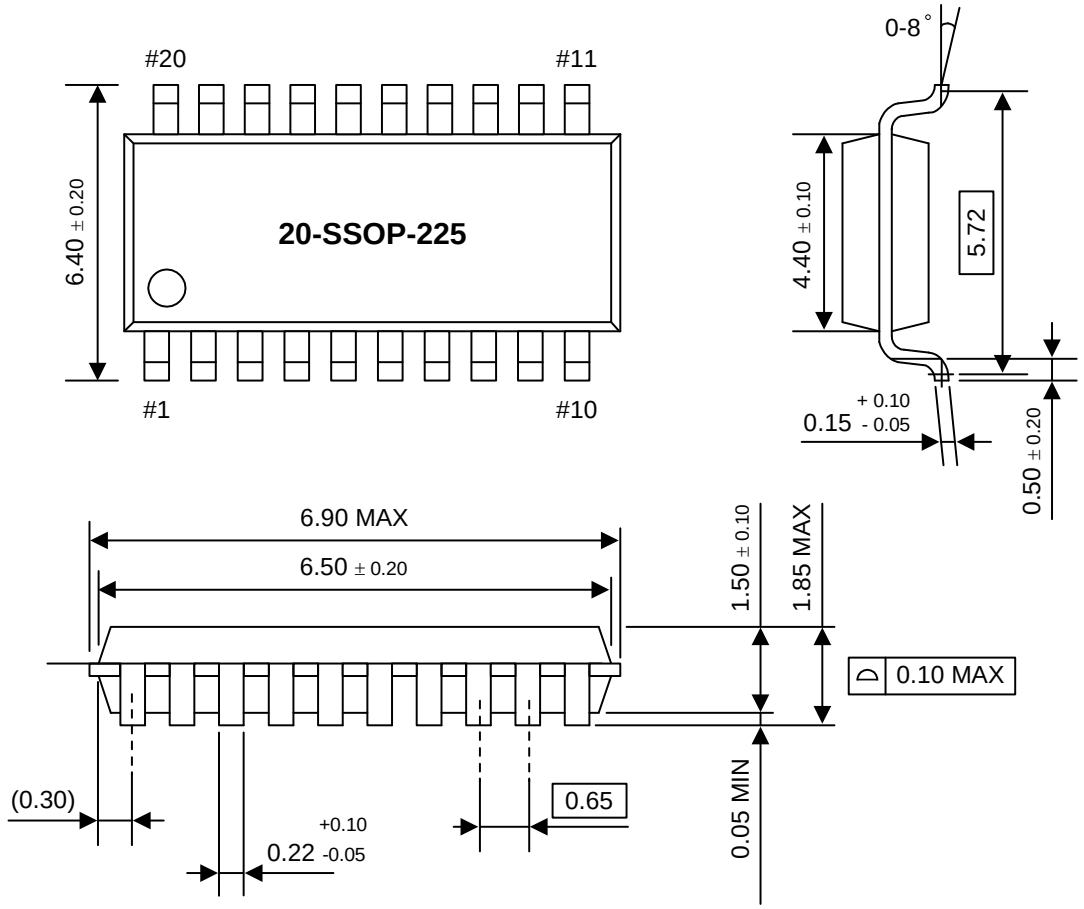
NOTE: Dimensions are in millimeters.

Figure 15-1 20-DIP-300A Package Dimensions



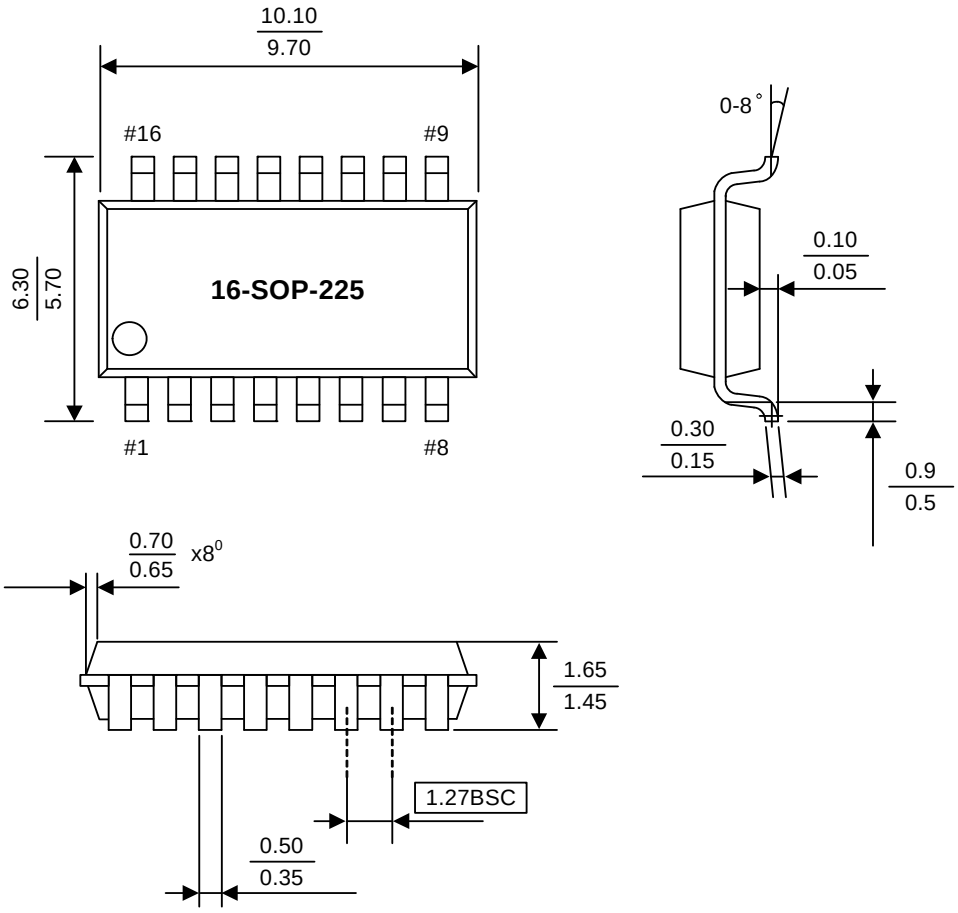
NOTE: Dimensions are in millimeters.

Figure 15-2 20-SOP-375 Package Dimensions



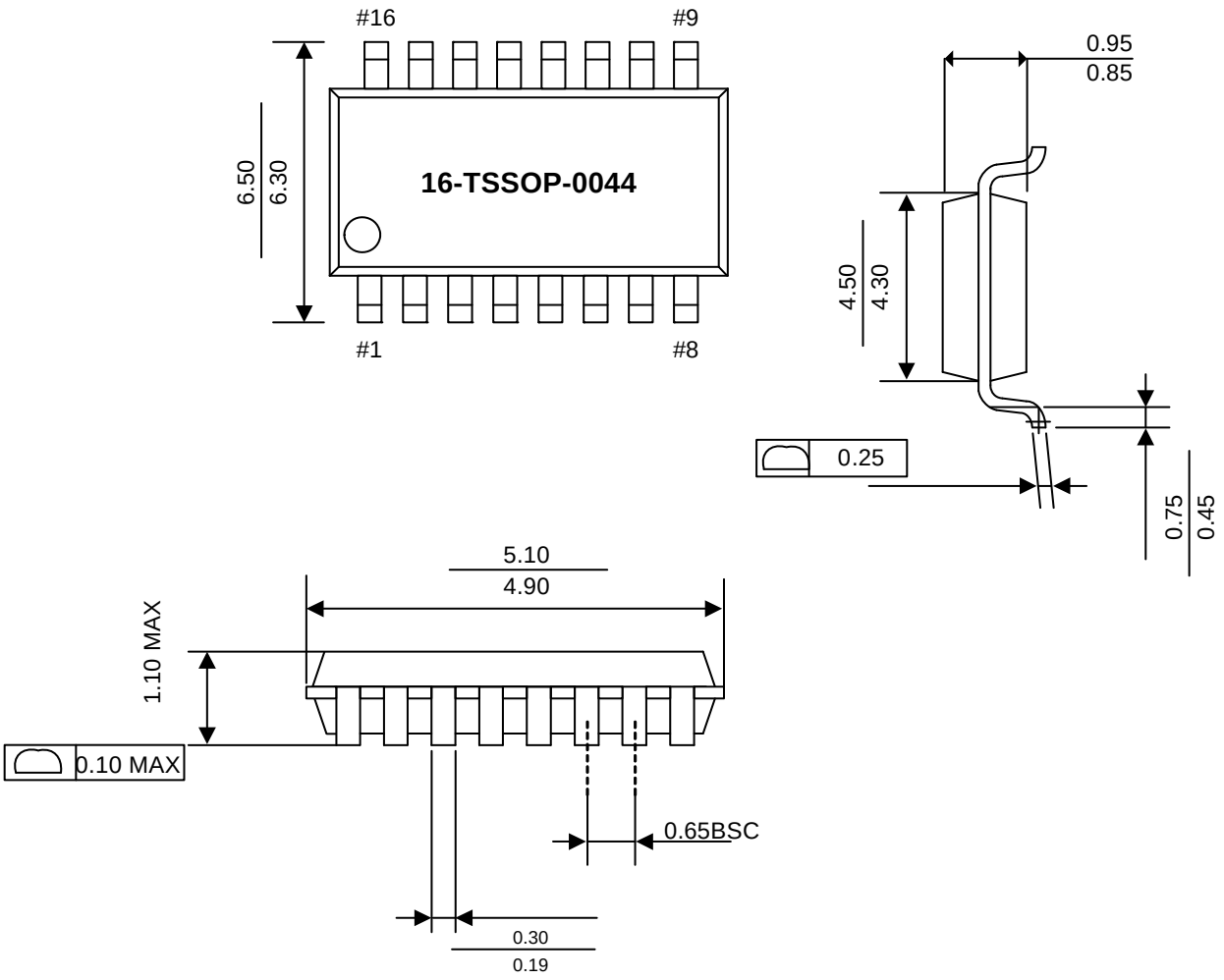
NOTE: Dimensions are in millimeters.

Figure 15-3 20-SSOP-225 Package Dimensions



NOTE: Dimensions are in millimeters.

Figure 15-4 16-SOP-225 Package Dimensions



NOTE: Dimensions are in millimeters.

Figure 15-5 16-TSSOP-0044 Package Dimensions

16 S3F94C8/F94C4 FLASH MCU

16.1 Overview of Flash MCU

The S3F94C8/F94C4 single-chip CMOS microcontroller is a Flash microcontroller unit (MCU) with on-chip ROM of 8K/4K bytes. The Flash ROM is accessed by serial data format.

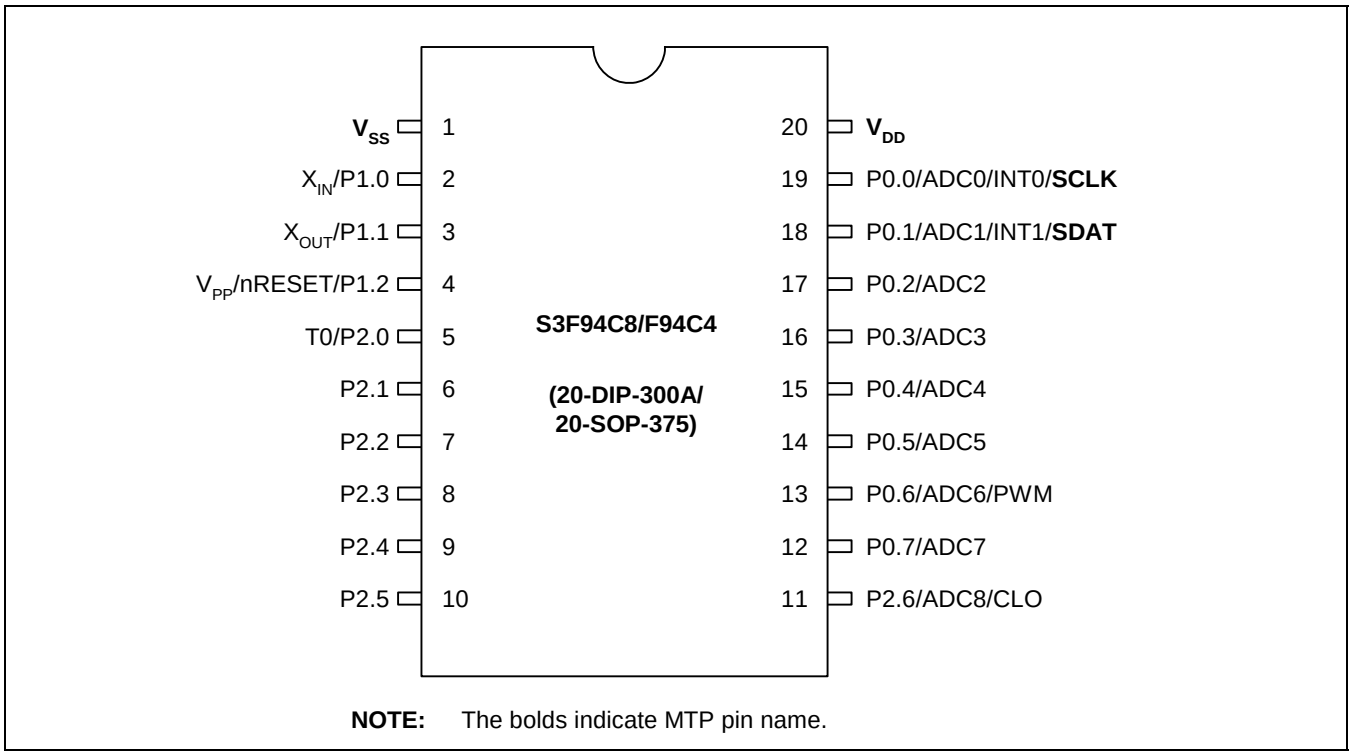
This serial data is transformed by two pins of the chip, namely, SCLK and SDAT. The SCLK pin specifies the synchronize signal. On the other hand, the SDAT pin facilitates data transfer using Flash Programmer Tool. The corresponding ports of SCLK and SDAT in S3F94C8/F94C4 are P0.0 and P1.1, respectively.

To enter the Flash Tool mode and to operate the chip, high power supply is required. Therefore, the VDD and VSS of chip must be connected to power and ground, respectively. The VPP port is reserved for providing high power during Flash operation. Its corresponding pin in S3F94C8/F94C4 is nRESET (P1.2) pin.

Table 16-1 shows the detailed description of the pin functions. *Figure 16-1* and *Figure 16-2* show the pin assignments of S3F94C8/F94C4 package types.

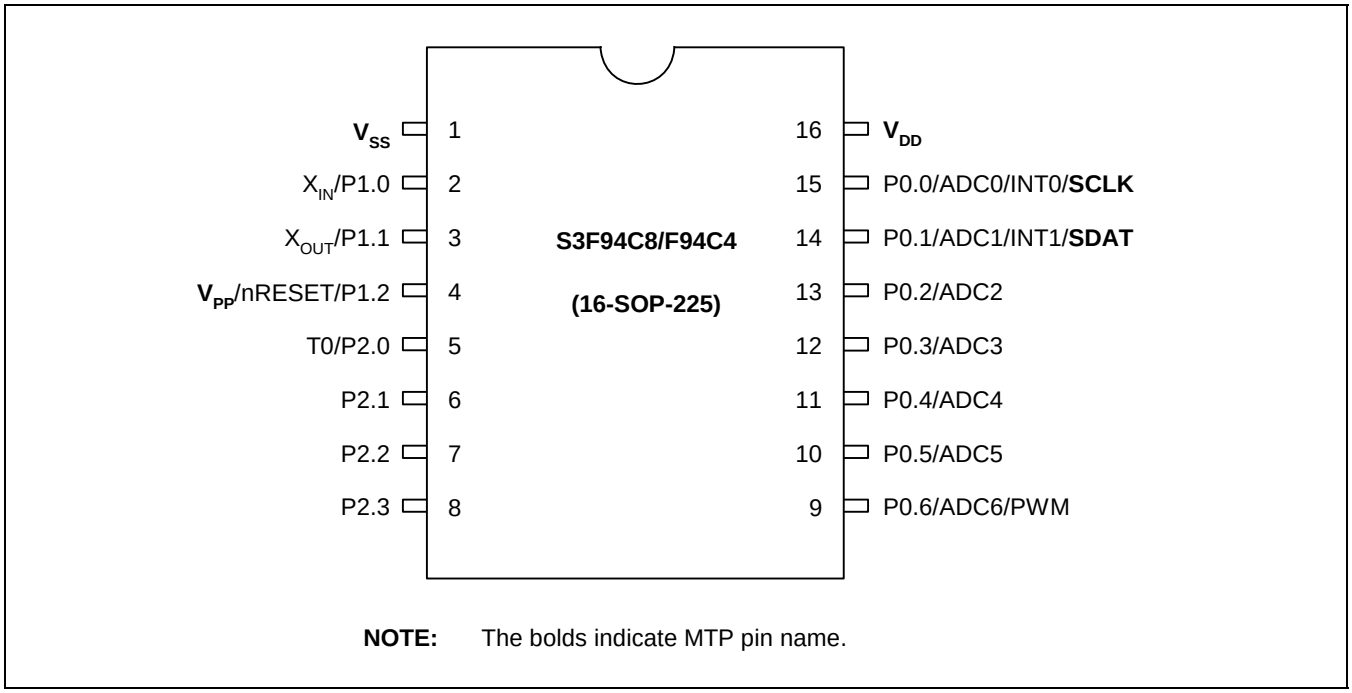
NOTE:

1. This chapter is about the Tool Program Mode of Flash MCU. If you want to know about User Program Mode, refer to the *Chapter 13, "Embedded Flash Memory Interface"*.
2. In S3F94C8/F94C4, only five pins are used as flash operation pins. The nRESET pin is used as VPP input and without TEST pin that different with other Samsung MCU products.



NOTE: The bolds indicate MTP pin name.

Figure 16-1 S3F94C8/F94C4 Pin Assignments (20-DIP/20SOP)



NOTE: The bolds indicate MTP pin name.

Figure 16-2 S3F94C8/F94C4 Pin Assignments (16SOP)

Table 16-1 Descriptions of Pins Used to Read/Write the EPROM

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P0.1	S_{DAT}	18 (20-pin) 14 (16-pin)	I/O	Serial data pin (output when reading, Input when writing) Input and push-pull output port can be assigned
P0.0	S_{CLK}	19 (20-pin) 15 (16-pin)	I	Serial clock pin (input only pin)
RESET/P1.2	V_{PP}	4	I	Power supply pin for entering Tool mode (indicates that MTP enters into the Tool mode). When 11 V is applied, MTP is in the Tool mode.
V_{DD}/V_{SS}	V_{DD}/V_{SS}	20 (20-pin), 16 (16-pin) 1 (20-pin), 1 (16-pin)	I	Logic power supply pin.

NOTE: Parentheses indicate pin number for 20-DIP-300A package.

Table 16-2 Comparison of S3F94C8/F94C4 Features

Characteristic	S3F94C8/F94C4
Program memory	8K/4K-byte Flash ROM
Operating voltage (V_{DD})	2.0 V to 5.5 V
Flash MCU programming mode	V_{DD} = 5.0 V, V_{PP} (nRESET) = 11 V
Pin configuration	20 DIP/20 SOP/20 SSOP /16SOP/16TSSOP
Programmability	User program multi-time

16.2 On Board Writing

The S3F94C8/F94C4 needs only five signal lines, including VDD and GND pins, for writing internal flash memory with serial protocol. Therefore, on-board writing is possible if the writing signal lines are considered during design of the PCB of application board.

16.2.1 Circuit Design Guide

At the flash writing, the writing tool needs five signal lines, including GND, VDD, VPP, SDAT, and SCLK. When you design the PCB circuit, consider the usage of these signal lines for on-board writing.

In case of VPP (nRESET) pin, a capacitor should be inserted between the VPP pin and GND to increase the noise effect.

Caution: Be careful while designing the related circuits of signal pins. The rising/falling timing of VPP, SCLK and SDAT is important for proper programming.

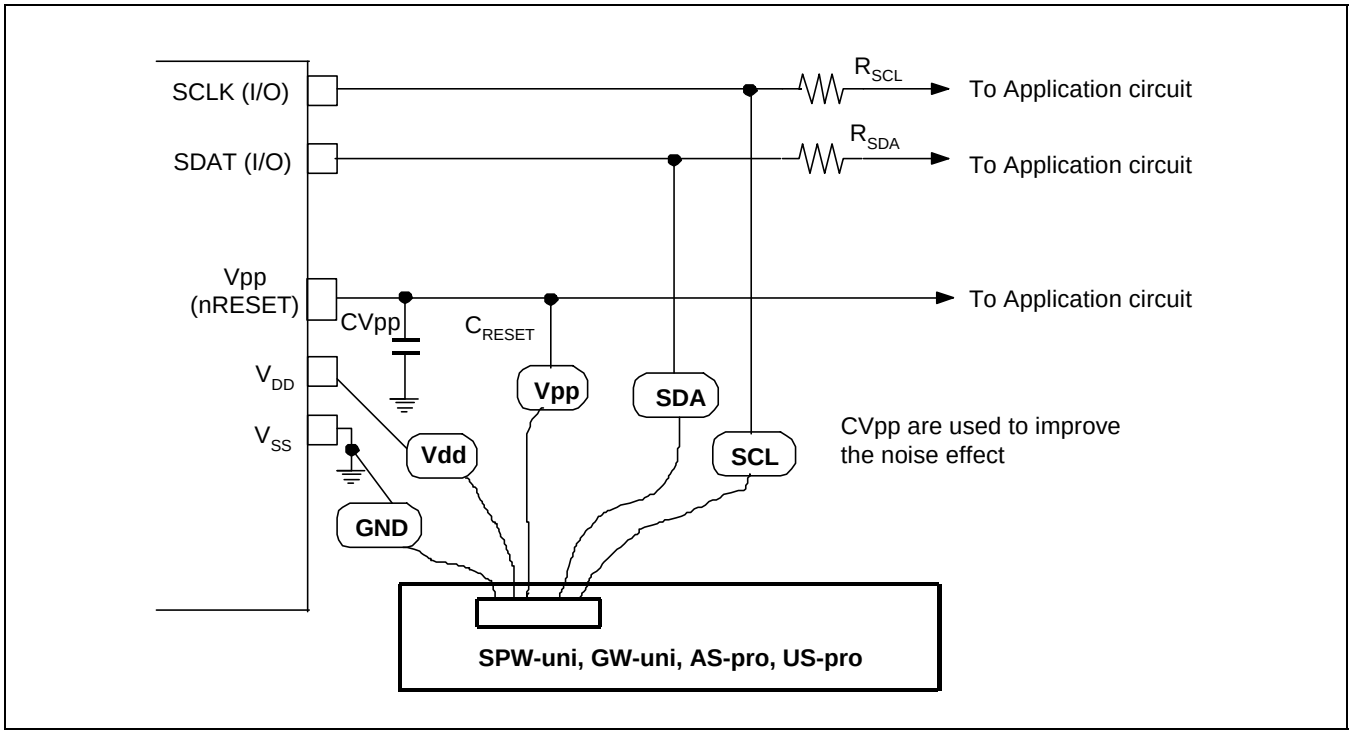


Figure 16-3 PCB design guide for on-board programming

Table 16-3 Reference Table for Connection

Pin Name	I/O mode in Applications	Resistor (need)	Required value
Vpp (nRESET)	Input	Yes	CVpp varies from 0.01uF ~ 0.02uF
SDAT (I/O)	Input	Yes	RSDAT varies from 2 Kohm ~ 5 Kohm
	Output	No (1)	-
SCLK (I/O)	Input	Yes	RSCLK varies from 2 Kohm ~ 5 Kohm
	Output	No (1)	-

NOTE:

1. In on-board writing mode, high-speed signal will be provided to SCLK and SDAT pins. If the application circuit is designed for handling high-speed response such as relay control circuit, high-speed signals may cause damage to the application circuits connected to SCLK or SDAT port. If possible, the I/O configuration of SDAT and SCLK pins must be set to input mode.
2. The value of R and C in this table specifies the recommended value. It varies with the circuit of system.

16.3 Information Block

The S3F94C8/94C4 provides a special flash area for storing chip ID or customer's information into it called the information block. This block is separate from the main flash ROM, which includes the flash ROM memory erase/write/read/read protection operation. The information in special flash area can be erased/written/read by Flash Programmer Tools, and is not available in the user mode.

The size of information block is 256Bytes. Since it is separate from flash ROM, the programming operation (chip erase/write) will not erase/change the data in information block. You can write the Chip ID (different for each chip) into it to distinguish every chip. This is useful for anti-imitation by storing production related information in this area.

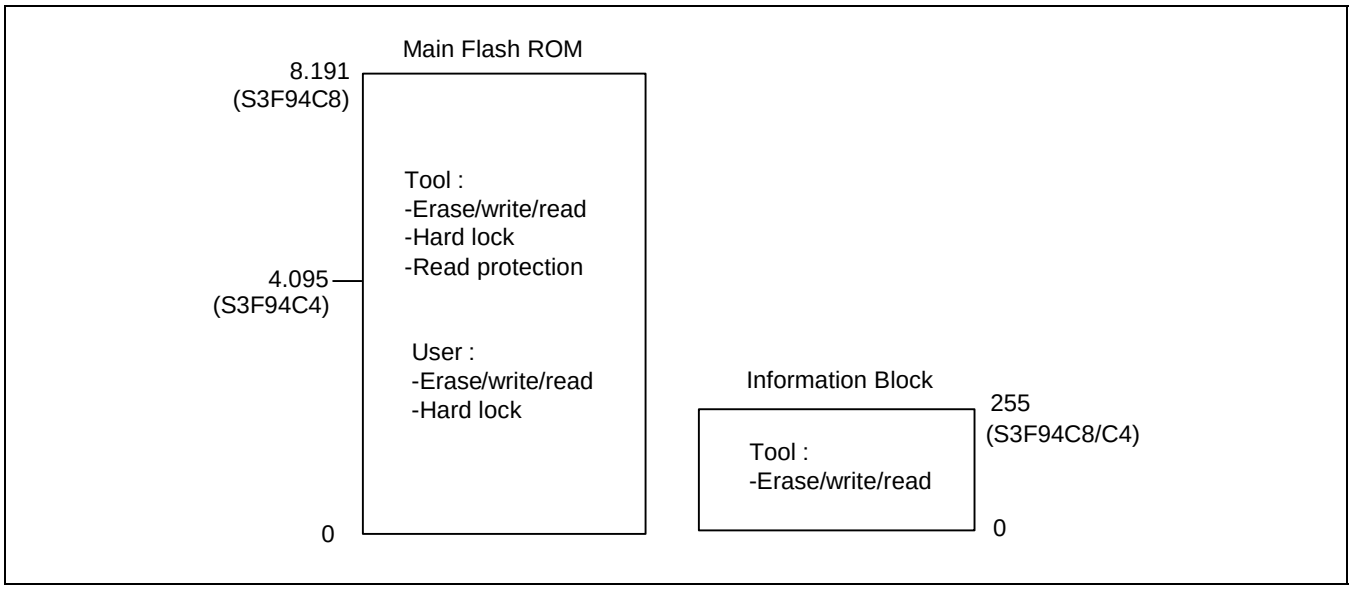


Figure 16-4 S3F94C8/F94C4 Flash Architecture

Table 16-4 Operation Results Comparison of Main Flash ROM and Information Blocks

Mode	Operation	Main Flash ROM	Information Block
Tool Mode	Erase MTP	Yes	No
	Program ROM / Read ROM	Yes	No
	Hard Lock / Read Protection	Yes	No
	Information Block Erase	No	Yes
	Information Block Write/Read	No	Yes
User Mode	Sector erase	Yes	No
	Write Byte /Read Byte	Yes	No
	Hard Lock	Yes	No

17 DEVELOPMENT TOOLS

17.1 Overview of Development Tools

Samsung provides a powerful and easy-to-use development support system on a turnkey basis. The development support system is composed of a host system, debugging tools, and supporting software. For a host system, any standard computer that employs Win95/98/2000/XP as its operating system can be used. A sophisticated debugging tool is provided in both the hardware and software. This tool includes an in-circuit emulator, OPENice-i500/i2000 and SK-1200 for the S3F7, S3F9, and S3F8 microcontroller families. Samsung also offers supporting software that includes debugger, assembler, and program for setting options.

17.1.1 Target Boards

Target boards are available for all the S3C9/S3F9 series microcontrollers. All the required target system cables and adapters are included in the device-specific target board. TB94C8/94C4 is a specific target board for the development of application systems using S3F94C8/F94C4.

17.1.2 Programming Socket Adapter

When you program S3F94C8/F94C4's flash memory by using an emulator or OTP/MTP writer, you need a specific programming socket adapter for S3F94C8/F94C4.

17.1.3 Development System Configuration

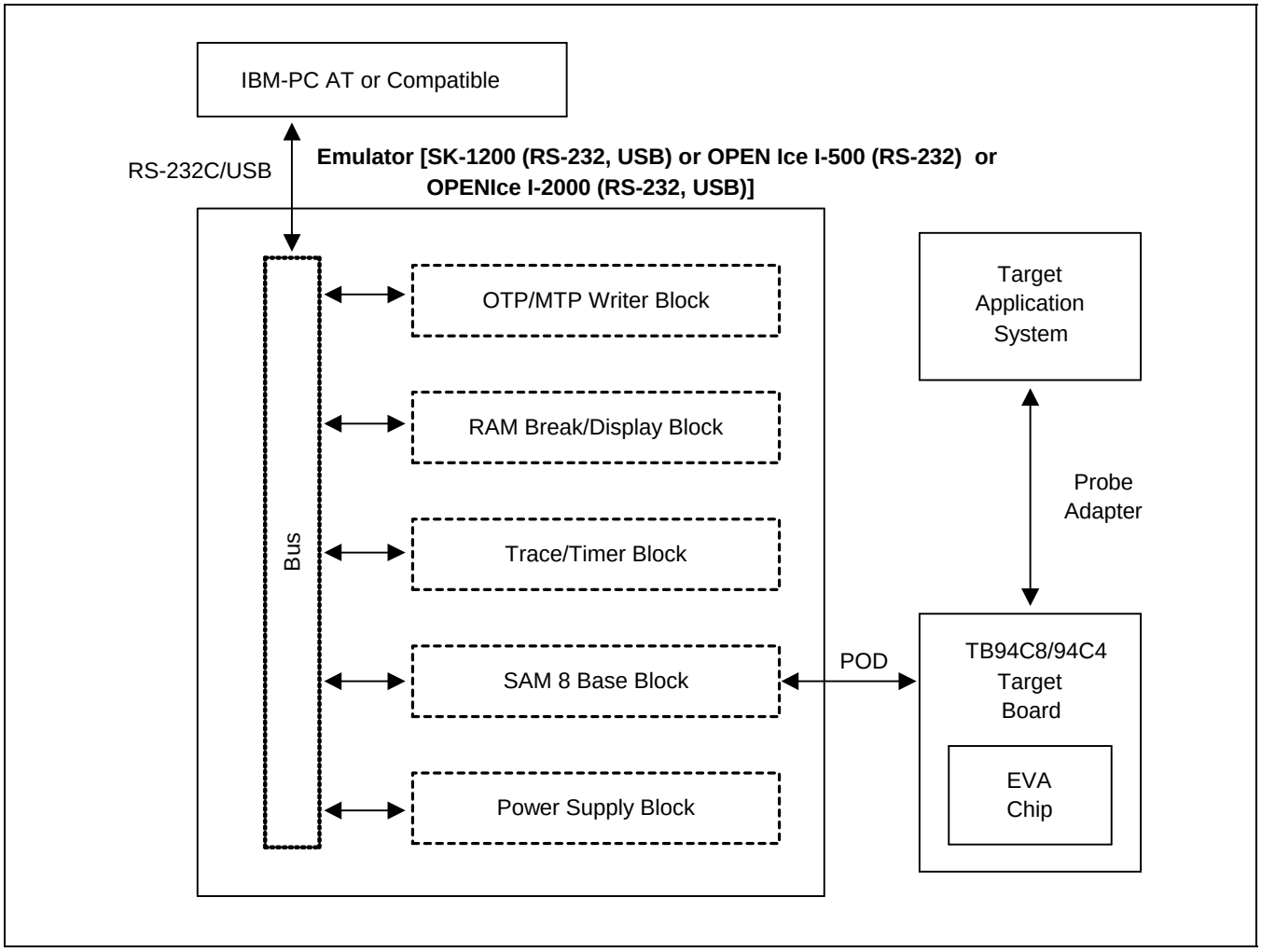


Figure 17-1 Development System Configuration

17.1.4 TB94C8/94C4 Target Board

The TB94C8/94C4 target board is used for S3F94C8/F94C4 microcontrollers. This board is operated as target CPU with Emulator (OPENIce I-500/2000, SK-1200).

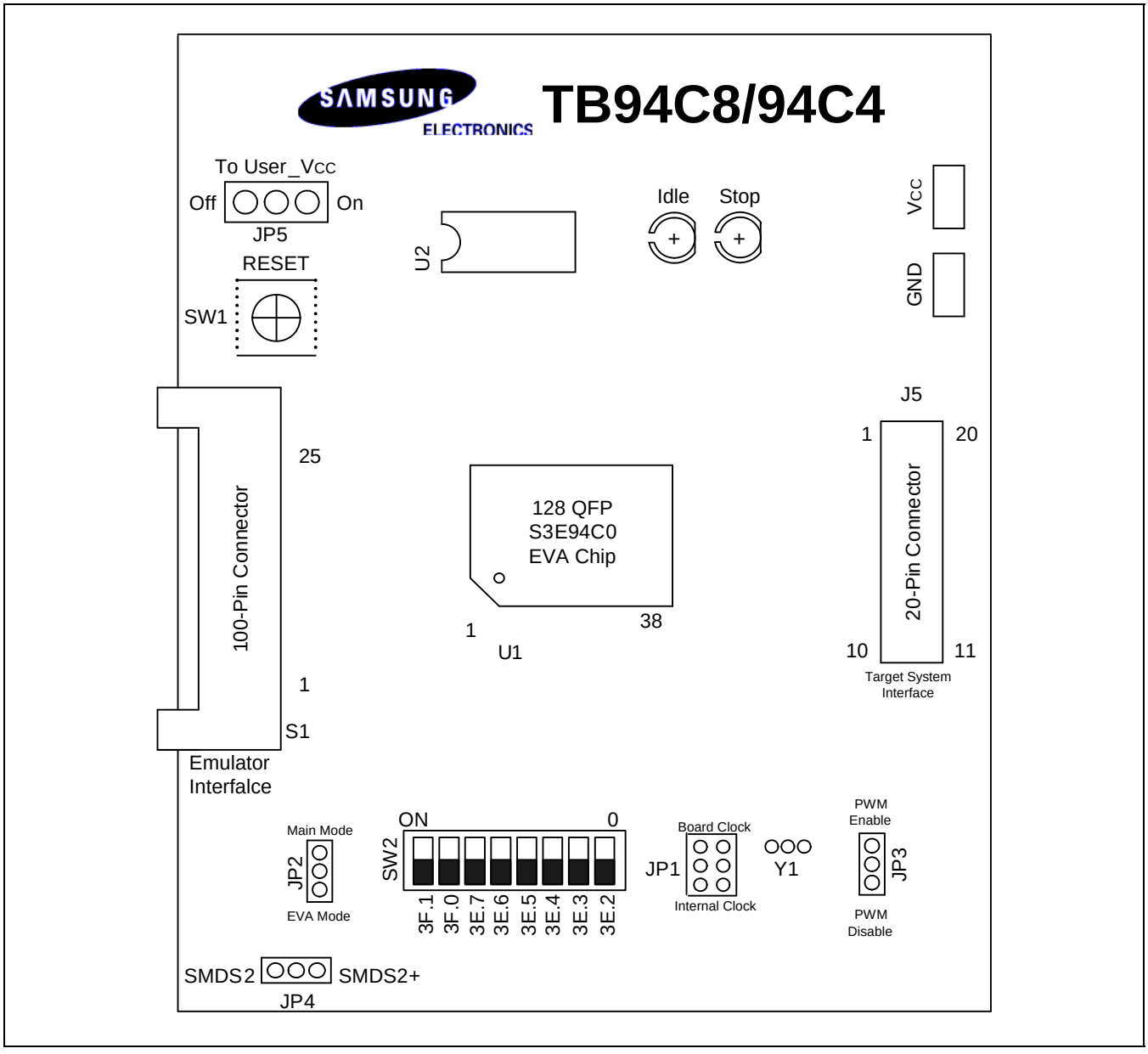


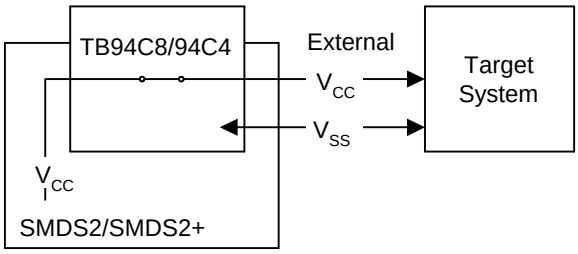
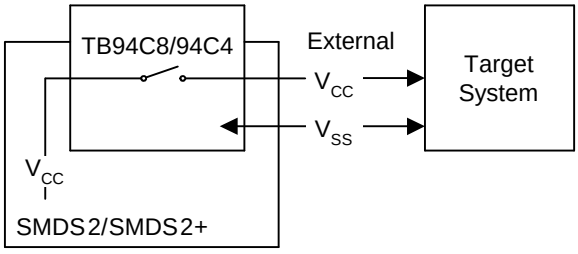
Figure 17-2 TB94C8/94C4 Target Board Configuration

NOTE: Since TB94C8/94C4 should be supplied with 5V, the power supply from Emulator should be 5V for the target board operation.

Table 17-1 Components of TB94C8/94C4

Symbols	Usage	Description
S1	100-pin connector	Connection between emulator and TB94C8/94C4 target board
J5	20-pin connector	Connection between target board and user application system
SW2	8-pin switch	Smart Option setting for S3F94C8/94C4 EVA-chip
RESET	Push button	Generates low active reset signal to S3F94C8/94C4 EVA-chip
VCC, GND	POWER connector	External power connector for TB94C8/94C4
IDLE, STOP LED	STOP/IDLE Display	Indicates the status of STOP or IDLE mode of S3F94C8/94C4 EVA-chip on TB94C8/94C4 target board
JP1	Clock Source Selection	Selection of SMDS2/SMDS2+ internal /external clock
JP2	MODE Selection	Selection of Eva/Main-chip mode of S3F94C8/94C4 EVA-chip
JP3	PWM selection	Enables/disables the selection of PWM
JP4	Emulator selection	Selection of SMDS2/SMDS2+
JP5	User's Power selection	Selection of power to user

Table 17-2 Power Selection Settings for TB94C8/94C4

"To User_Vcc" Settings	Operating Mode	Comments
To user_Vcc off on		The SMDS2/SMDS2+ main board supplies V_{CC} to the target board (evaluation chip) and target system.
To user_Vcc off on		The SMDS2/SMDS2+ main board supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

NOTE: The symbol in the "To User_Vcc" setting column indicates the electrical short (off) configuration:

17.1.5 SMDS2+ Selection (SAM8)

To write data into program memory available in SMDS2+, the target board should be selected for SMDS2+ through a switch (as shown in *Table 17-3*). Otherwise, the program memory writing function is not available.

Table 17-3 The SMDS2+ Tool Selection Setting

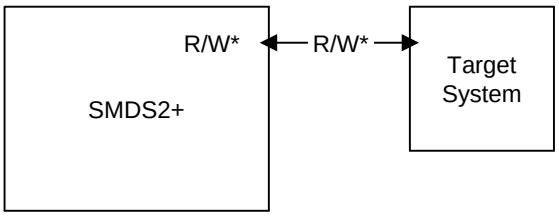
"JP4" Setting	Operating Mode
SMDS 2 SMDS 2+	

Table 17-4 Using Single Header Pins to Select Clock Source / PWM / Operation Mode

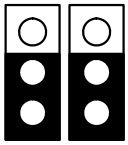
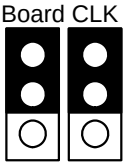
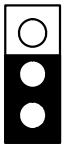
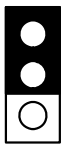
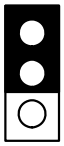
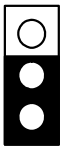
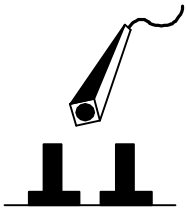
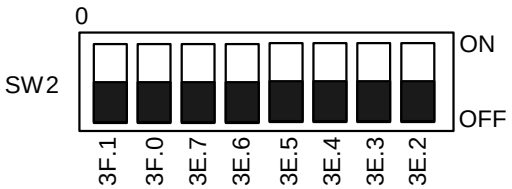
Target Board Part	Comments
Board CLK  JP1 Clock Source Inner CLK	Use SMDS2/SMDS2+ internal clock source as the system clock (Default setting).
Board CLK  JP1 Clock Source Inner CLK	Use external crystal or ceramic oscillator as the system clock.
PWM Enable  JP3 PWM Disable	PWM function is disabled.
PWM Enable  JP3 PWM Disable	PWM function is enabled (Default setting).
Main Mode  JP2 EVA Mode	S3E94C0 runs in the Main mode, similar to S3F94C8/F94C4. The debug interface is not available.
Main Mode  JP2 EVA Mode	S3E94C0 runs in the EVA mode. While debugging a program, set the jumper in this mode (Default setting).

Table 17-5 Using Single Header Pins as the Input Path for External Trigger Sources

Target Board Part	Comments
External Triggers Ch1(TP3) Ch2(TP4)	 Connector from External Trigger Sources of the Application System You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for SK-1000/SMDS2+ breakpoint and trace functions.



ON	Low
OFF	High (Default)

NOTE:
1. For EVA chip, smart option is determined by DIP switch not software.
2. Please keep the reserved bits as default value (high).

Figure 17-3 DIP Switch for Smart Option

- IDLE LED
  The IDLE LED is ON when the evaluation chip (S3E94C0) is in the idle mode.
- STOP LED
  The STOP LED is ON when the evaluation chip (S3E94C0) is in the stop mode.

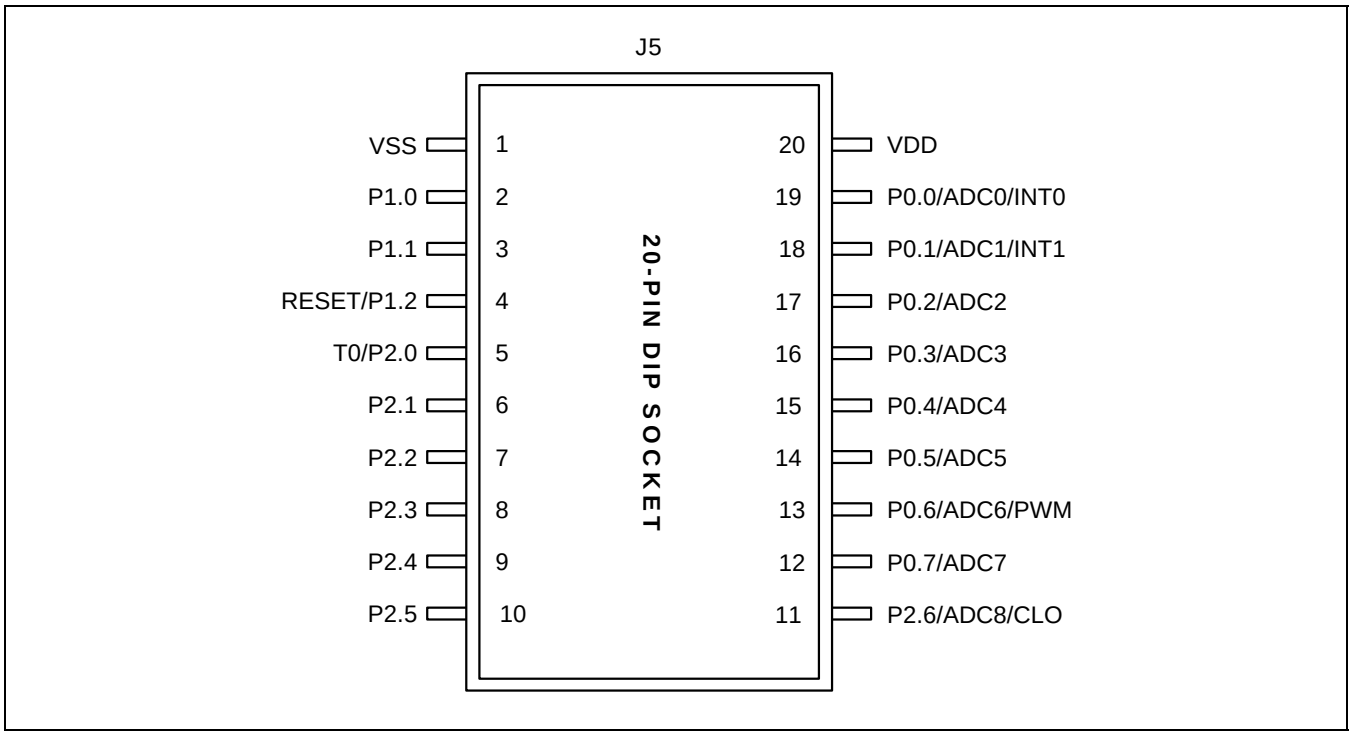


Figure 17-4 20-Pin Connector for TB94C8/94C4

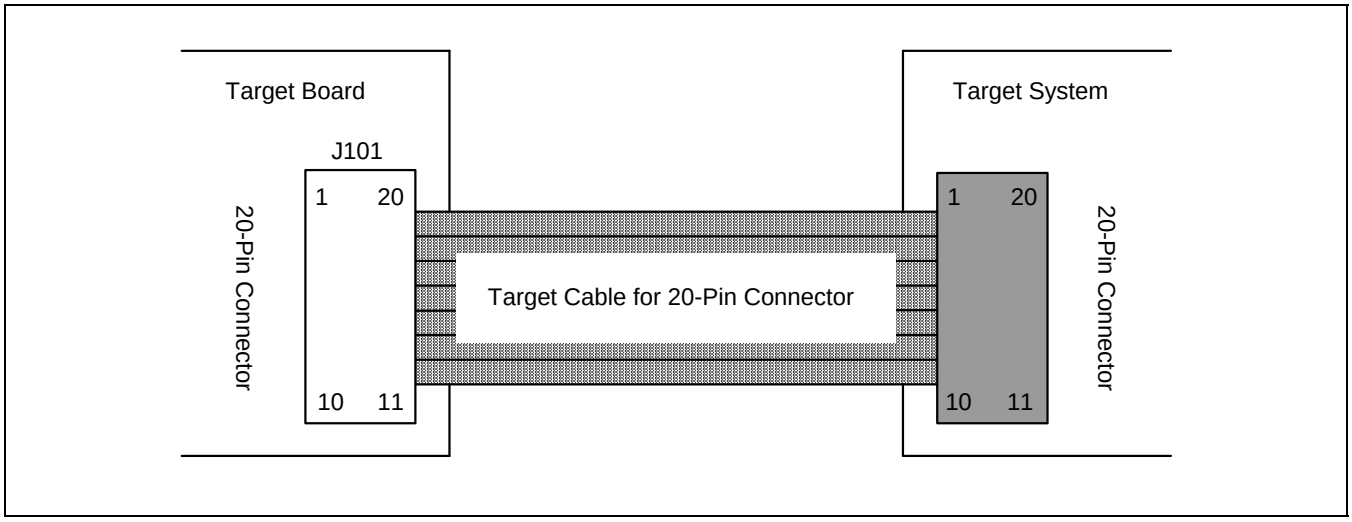


Figure 17-5 S3F94C8/F94C4 Probe Adapter for 20-DIP Package

17.1.6 Third parties for Development Tools

Samsung provides a complete line of development tools for its microcontrollers. The in-circuit emulator solution from Samsung covers a wide range of capabilities and prices, from a low cost ICE to a complete system, with an OTP/MTP programmer.

17.1.6.1 In-Circuit Emulator for SAM8 family

- OPENice-i500/2000
- SmartKit SK-1200

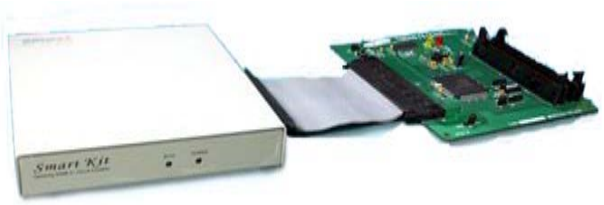
17.1.6.2 OTP/MTP Programmer

- SPW-uni
- GW-uni (8-gang programmer)
- AS-pro

17.1.6.3 Development Tools Suppliers

Contact our local sales offices or third party tool suppliers directly for getting these development tools.

17.1.6.4 8-bit In-Circuit Emulator

OPENice - i500 	AIJI System Telephone: 82-31-223-6611 Fax: 82-331-223-6613 E-mail: openice@aijisystem.com, stroh@yicsystem.com URL: http://www.aijisystem.com
OPENice - i2000 	AIJI System Telephone: 82-31-223-6611 Fax: 82-331-223-6613 E-mail: openice@aijisystem.com, stroh@yicsystem.com URL: http://www.aijisystem.com
SK-1200 	Seminix Telephone: 82-2-539-7891 Fax: 82-2-539-7819 E-mail: sales@seminix.com URL: http://www.seminix.com

OTP/MTP Programmer (Writer)

	SPW-uni Single OTP/ MTP/FLASH Programmer Supports Download/Upload and Data Edit functions Supports PC-based operation with USB port Supports full functions of OTP/MTP/FLASH MCU programmer (Read, Program, Verify, Blank, and Protection) Fast programming speed (4Kbyte/sec) Supports all Samsung OTP/MTP/FLASH MCU devices Low cost NOR Flash memory (SST, Samsung) NAND Flash memory (SLC) Supports new devices by adding device files or upgrading the software	SEMINIX Telephone: 82-2-539-7891 Fax: 82-2-539-7819 E-mail: sales@seminix.com URL: http://www.seminix.com
	GW-uni Gang Programmer for OTP/MTP/FLASH MCU Supports eight devices' programming at one time Fast programming speed: OTP (2Kbps)/ MTP (10Kbps) Maximum buffer memory: 100Mbyte Operation mode: PC based/ Stand-alone (no PC) Supports full functions of OTP/MTP (Read, Program, Checksum, Verify, Erase, Read protection, and Smart option) Simple Graphical User Interface (GUI) Sets device information using device part number Supports LCD display and touch key (Stand-alone mode operation) System upgradable (Simple firmware upgrade)	SEMINIX Telephone: 82-2-539-7891 Fax: 82-2-539-7819 E-mail: sales@seminix.com URL: http://www.seminix.com

OTP/MTP Programmer (Writer) (Continued)

	AS-pro On-board programmer for Samsung Flash MCU Portable and Standalone Samsung OTP/MTP/FLASH Programmer for after service Small size and light for portable use Supports all Samsung OTP/MTP/FLASH devices Supports HEX file download via USB port from PC Fast program and verify time (OTP: 2Kbytes per second, MTP: 10Kbytes per second) Internal large buffer memory (118M Bytes) Driver software runs on various operating systems (Windows 95/98/2000/XP) Supports full functions of OTP/MTP programmer (Read, Program, Verify, Blank, and Protection) Supplies two kinds of Power (User system power or USB power adapter) Supports Firmware upgrade	SEMINIX Telephone: 82-2-539-7891 Fax: 82-2-539-7819 E-mail: sales@seminix.com URL: http://www.seminix.com
	Flash writing adapter board Special flash writing socket for S3F94C8/F94C4 Supports 20DIP, 20SOP, 20SSOP, 16DIP, 16SOP, and 16TSSOP	C&A technology Telephone: 82-2-2612-9027 Fax: 82-2-2612-9044 E-mail: wisdom@cnatech.com URL: http://www.cnatech.com